

libstdc++

Generated by Doxygen 1.6.1

Fri Mar 12 10:49:44 2010

Contents

1	Todo List	1
2	Module Documentation	11
2.1	Extensions	11
2.1.1	Detailed Description	11
2.2	SGI STL extensions	12
2.2.1	Detailed Description	15
2.2.2	Function Documentation	16
2.2.2.1	__median	16
2.2.2.2	__median	16
2.2.2.3	_Find_first	17
2.2.2.4	_Find_next	17
2.2.2.5	_Unchecked_set	18
2.2.2.6	compose1	18
2.2.2.7	compose2	18
2.2.2.8	constant0	18
2.2.2.9	constant1	18
2.2.2.10	constant2	18
2.2.2.11	copy_n	19
2.2.2.12	distance	19
2.2.2.13	identity_element	20
2.2.2.14	identity_element	20
2.2.2.15	iota	20

2.2.2.16	is_heap	20
2.2.2.17	is_heap	20
2.2.2.18	is_sorted	21
2.2.2.19	is_sorted	21
2.2.2.20	lexicographical_compare_3way	21
2.2.2.21	power	22
2.2.2.22	power	22
2.2.2.23	random_sample	22
2.2.2.24	random_sample	23
2.2.2.25	random_sample_n	23
2.2.2.26	random_sample_n	23
2.2.2.27	uninitialized_copy_n	24
2.3	Containers	25
2.3.1	Detailed Description	25
2.4	Sequences	26
2.4.1	Detailed Description	27
2.5	Associative Containers	28
2.5.1	Detailed Description	28
2.6	Unordered Associative Containers	29
2.6.1	Detailed Description	29
2.7	Diagnostics	30
2.7.1	Detailed Description	30
2.8	Concurrency	31
2.8.1	Detailed Description	31
2.9	Exceptions	32
2.9.1	Detailed Description	34
2.9.2	Typedef Documentation	34
2.9.2.1	terminate_handler	34
2.9.2.2	unexpected_handler	34
2.9.3	Function Documentation	35
2.9.3.1	__verbose_terminate_handler	35

2.9.3.2	copy_exception	35
2.9.3.3	current_exception	35
2.9.3.4	rethrow_exception	35
2.9.3.5	rethrow_if_nested	35
2.9.3.6	rethrow_if_nested	35
2.9.3.7	set_terminate	36
2.9.3.8	set_unexpected	36
2.9.3.9	terminate	36
2.9.3.10	throw_with_nested	36
2.9.3.11	uncaught_exception	36
2.9.3.12	unexpected	36
2.10	Time	37
2.10.1	Detailed Description	37
2.11	Complex Numbers	38
2.11.1	Detailed Description	42
2.11.2	Function Documentation	42
2.11.2.1	abs	42
2.11.2.2	acos	43
2.11.2.3	acosh	43
2.11.2.4	arg	43
2.11.2.5	arg	43
2.11.2.6	asin	43
2.11.2.7	asinh	44
2.11.2.8	atan	44
2.11.2.9	atanh	44
2.11.2.10	conj	44
2.11.2.11	cos	44
2.11.2.12	cosh	44
2.11.2.13	exp	45
2.11.2.14	fabs	45
2.11.2.15	log	45

2.11.2.16 <code>log10</code>	45
2.11.2.17 <code>norm</code>	45
2.11.2.18 <code>operator!=</code>	46
2.11.2.19 <code>operator!=</code>	46
2.11.2.20 <code>operator!=</code>	46
2.11.2.21 <code>operator*</code>	46
2.11.2.22 <code>operator*</code>	46
2.11.2.23 <code>operator*</code>	46
2.11.2.24 <code>operator*=</code>	47
2.11.2.25 <code>operator*=</code>	47
2.11.2.26 <code>operator+</code>	47
2.11.2.27 <code>operator+</code>	47
2.11.2.28 <code>operator+</code>	47
2.11.2.29 <code>operator+</code>	47
2.11.2.30 <code>operator+=</code>	48
2.11.2.31 <code>operator-</code>	48
2.11.2.32 <code>operator-</code>	48
2.11.2.33 <code>operator-</code>	48
2.11.2.34 <code>operator-</code>	48
2.11.2.35 <code>operator-=</code>	48
2.11.2.36 <code>operator/</code>	49
2.11.2.37 <code>operator/</code>	49
2.11.2.38 <code>operator/</code>	49
2.11.2.39 <code>operator/=</code>	49
2.11.2.40 <code>operator/=</code>	49
2.11.2.41 <code>operator<<</code>	49
2.11.2.42 <code>operator=</code>	50
2.11.2.43 <code>operator=</code>	50
2.11.2.44 <code>operator==</code>	50
2.11.2.45 <code>operator==</code>	50
2.11.2.46 <code>operator==</code>	50

2.11.2.47	operator>>	51
2.11.2.48	polar	51
2.11.2.49	pow	51
2.11.2.50	pow	51
2.11.2.51	pow	51
2.11.2.52	sin	52
2.11.2.53	sinh	52
2.11.2.54	sqrt	52
2.11.2.55	tan	52
2.11.2.56	tanh	52
2.12	Condition Variables	53
2.12.1	Detailed Description	53
2.12.2	Enumeration Type Documentation	53
2.12.2.1	cv_status	53
2.13	Futures	54
2.13.1	Detailed Description	56
2.13.2	Enumeration Type Documentation	56
2.13.2.1	future_errc	56
2.13.3	Variable Documentation	56
2.13.3.1	future_category	56
2.14	I/O	57
2.14.1	Detailed Description	59
2.14.2	Typedef Documentation	60
2.14.2.1	filebuf	60
2.14.2.2	fstream	60
2.14.2.3	ifstream	60
2.14.2.4	ios	60
2.14.2.5	iostream	60
2.14.2.6	istream	61
2.14.2.7	istringstream	61
2.14.2.8	ofstream	61

2.14.2.9	ostream	61
2.14.2.10	ostreamstream	61
2.14.2.11	streambuf	61
2.14.2.12	stringbuf	61
2.14.2.13	stringstream	62
2.14.2.14	wfilebuf	62
2.14.2.15	wfstream	62
2.14.2.16	wifstream	62
2.14.2.17	wios	62
2.14.2.18	wiostream	62
2.14.2.19	wistream	62
2.14.2.20	wstringstream	63
2.14.2.21	wofstream	63
2.14.2.22	wostream	63
2.14.2.23	wstringstream	63
2.14.2.24	wstreambuf	63
2.14.2.25	wstringbuf	63
2.14.2.26	wstringstream	63
2.15	Memory	64
2.15.1	Detailed Description	64
2.16	Pointer Abstractions	65
2.16.1	Detailed Description	67
2.16.2	Function Documentation	67
2.16.2.1	allocate_shared	67
2.16.2.2	get_deleter	67
2.16.2.3	make_shared	68
2.16.2.4	operator<<	68
2.17	Mutexes	69
2.17.1	Detailed Description	70
2.17.2	Function Documentation	70
2.17.2.1	call_once	70

2.17.2.2	lock	70
2.17.2.3	try_lock	71
2.18	Numerics	72
2.18.1	Detailed Description	72
2.19	Rational Arithmetic	73
2.19.1	Detailed Description	74
2.20	Threads	75
2.20.1	Detailed Description	75
2.21	Utilities	76
2.21.1	Detailed Description	76
2.22	Numeric Arrays	77
2.22.1	Detailed Description	88
2.22.2	Function Documentation	89
2.22.2.1	gslice	89
2.22.2.2	gslice	89
2.22.2.3	gslice	89
2.22.2.4	gslice_array	89
2.22.2.5	indirect_array	89
2.22.2.6	mask_array	90
2.22.2.7	slice	90
2.22.2.8	slice	90
2.22.2.9	slice_array	90
2.22.2.10	valarray	90
2.22.2.11	valarray	90
2.22.2.12	valarray	91
2.22.2.13	valarray	91
2.22.2.14	valarray	91
2.22.2.15	valarray	91
2.22.2.16	valarray	91
2.22.2.17	valarray	91
2.22.2.18	valarray	92

2.22.2.19 <code>~gslice</code>	92
2.22.2.20 <code>apply</code>	92
2.22.2.21 <code>apply</code>	92
2.22.2.22 <code>cshift</code>	93
2.22.2.23 <code>max</code>	93
2.22.2.24 <code>min</code>	93
2.22.2.25 <code>operator!</code>	93
2.22.2.26 <code>operator%=</code>	94
2.22.2.27 <code>operator%=</code>	94
2.22.2.28 <code>operator%=</code>	94
2.22.2.29 <code>operator%=</code>	94
2.22.2.30 <code>operator%=</code>	94
2.22.2.31 <code>operator%=</code>	94
2.22.2.32 <code>operator&=</code>	95
2.22.2.33 <code>operator&=</code>	95
2.22.2.34 <code>operator&=</code>	95
2.22.2.35 <code>operator&=</code>	95
2.22.2.36 <code>operator&=</code>	95
2.22.2.37 <code>operator&=</code>	96
2.22.2.38 <code>operator*=</code>	96
2.22.2.39 <code>operator*=</code>	96
2.22.2.40 <code>operator*=</code>	96
2.22.2.41 <code>operator*=</code>	96
2.22.2.42 <code>operator*=</code>	96
2.22.2.43 <code>operator*=</code>	97
2.22.2.44 <code>operator+</code>	97
2.22.2.45 <code>operator+=</code>	97
2.22.2.46 <code>operator+=</code>	97
2.22.2.47 <code>operator+=</code>	97
2.22.2.48 <code>operator+=</code>	97
2.22.2.49 <code>operator+=</code>	98

2.22.2.50 operator+=	98
2.22.2.51 operator-	98
2.22.2.52 operator-=	98
2.22.2.53 operator-=	98
2.22.2.54 operator-=	99
2.22.2.55 operator-=	99
2.22.2.56 operator-=	99
2.22.2.57 operator-=	99
2.22.2.58 operator/=	99
2.22.2.59 operator/=	99
2.22.2.60 operator/=	100
2.22.2.61 operator/=	100
2.22.2.62 operator/=	100
2.22.2.63 operator/=	100
2.22.2.64 operator<<=	100
2.22.2.65 operator<<=	100
2.22.2.66 operator<<=	101
2.22.2.67 operator<<=	101
2.22.2.68 operator<<=	101
2.22.2.69 operator<<=	101
2.22.2.70 operator=	101
2.22.2.71 operator=	101
2.22.2.72 operator=	102
2.22.2.73 operator=	102
2.22.2.74 operator=	102
2.22.2.75 operator=	102
2.22.2.76 operator=	102
2.22.2.77 operator=	103
2.22.2.78 operator=	103
2.22.2.79 operator=	103
2.22.2.80 operator=	103

2.22.2.81 operator=	103
2.22.2.82 operator=	104
2.22.2.83 operator=	104
2.22.2.84 operator=	104
2.22.2.85 operator=	104
2.22.2.86 operator=	105
2.22.2.87 operator=	105
2.22.2.88 operator=	105
2.22.2.89 operator>>=	106
2.22.2.90 operator>>=	106
2.22.2.91 operator>>=	106
2.22.2.92 operator>>=	106
2.22.2.93 operator>>=	106
2.22.2.94 operator>>=	106
2.22.2.95 operator[]	107
2.22.2.96 operator[]	107
2.22.2.97 operator[]	107
2.22.2.98 operator[]	108
2.22.2.99 operator[]	108
2.22.2.100operator[]	109
2.22.2.101operator[]	109
2.22.2.102operator[]	110
2.22.2.103operator[]	110
2.22.2.104operator^=	110
2.22.2.105operator^=	111
2.22.2.106operator^=	111
2.22.2.107operator^=	111
2.22.2.108operator^=	111
2.22.2.109operator^=	111
2.22.2.110operator =	111
2.22.2.111operator =	112

2.22.2.112operator =	112
2.22.2.113operator =	112
2.22.2.114operator =	112
2.22.2.115operator =	112
2.22.2.116operator~	113
2.22.2.117resize	113
2.22.2.118shift	113
2.22.2.119size	113
2.22.2.120size	114
2.22.2.121size	114
2.22.2.122start	114
2.22.2.123start	114
2.22.2.124stride	114
2.22.2.125stride	114
2.22.2.126sum	115
2.23 Mathematical Special Functions	116
2.23.1 Detailed Description	119
2.23.2 Function Documentation	119
2.23.2.1 assoc_laguerre	119
2.23.2.2 assoc_legendre	119
2.23.2.3 beta	119
2.23.2.4 comp_ellint_1	119
2.23.2.5 comp_ellint_2	119
2.23.2.6 comp_ellint_3	120
2.23.2.7 conf_hyperg	120
2.23.2.8 cyl_bessel_i	120
2.23.2.9 cyl_bessel_j	120
2.23.2.10 cyl_bessel_k	120
2.23.2.11 cyl_neumann	120
2.23.2.12 ellint_1	121
2.23.2.13 ellint_2	121

2.23.2.14	ellint_3	121
2.23.2.15	expint	121
2.23.2.16	hermite	121
2.23.2.17	hyperg	122
2.23.2.18	laguerre	122
2.23.2.19	legendre	122
2.23.2.20	riemann_zeta	122
2.23.2.21	sph_bessel	122
2.23.2.22	sph_legendre	122
2.23.2.23	sph_neumann	123
2.24	Regular Expressions	124
2.24.1	Detailed Description	130
2.24.2	Typedef Documentation	130
2.24.2.1	cregex_token_iterator	130
2.24.2.2	csub_match	130
2.24.2.3	regex	130
2.24.2.4	sregex_token_iterator	131
2.24.2.5	ssub_match	131
2.24.2.6	wcregex_token_iterator	131
2.24.2.7	wsub_match	131
2.24.2.8	wregex	131
2.24.2.9	wsregex_token_iterator	131
2.24.2.10	wssub_match	131
2.24.3	Function Documentation	132
2.24.3.1	isctype	132
2.24.3.2	operator!=	132
2.24.3.3	operator!=	132
2.24.3.4	operator!=	133
2.24.3.5	operator!=	133
2.24.3.6	operator!=	134
2.24.3.7	operator!=	134

2.24.3.8	operator!=	134
2.24.3.9	operator!=	135
2.24.3.10	operator<	135
2.24.3.11	operator<	136
2.24.3.12	operator<	136
2.24.3.13	operator<	136
2.24.3.14	operator<	137
2.24.3.15	operator<	137
2.24.3.16	operator<	138
2.24.3.17	operator<<	138
2.24.3.18	operator<=	138
2.24.3.19	operator<=	139
2.24.3.20	operator<=	139
2.24.3.21	operator<=	139
2.24.3.22	operator<=	140
2.24.3.23	operator<=	140
2.24.3.24	operator<=	141
2.24.3.25	operator==	141
2.24.3.26	operator==	141
2.24.3.27	operator==	142
2.24.3.28	operator==	142
2.24.3.29	operator==	142
2.24.3.30	operator==	143
2.24.3.31	operator==	143
2.24.3.32	operator==	144
2.24.3.33	operator>	144
2.24.3.34	operator>	144
2.24.3.35	operator>	145
2.24.3.36	operator>	145
2.24.3.37	operator>	146
2.24.3.38	operator>	146

2.24.3.39	operator>	146
2.24.3.40	operator>=	147
2.24.3.41	operator>=	147
2.24.3.42	operator>=	148
2.24.3.43	operator>=	148
2.24.3.44	operator>=	148
2.24.3.45	operator>=	149
2.24.3.46	operator>=	149
2.24.3.47	regex_match	150
2.24.3.48	regex_match	150
2.24.3.49	regex_match	151
2.24.3.50	regex_match	152
2.24.3.51	regex_match	152
2.24.3.52	regex_match	153
2.24.3.53	regex_replace	154
2.24.3.54	regex_replace	154
2.24.3.55	regex_search	155
2.24.3.56	regex_search	156
2.24.3.57	regex_search	156
2.24.3.58	regex_search	157
2.24.3.59	regex_search	158
2.24.3.60	regex_search	158
2.24.3.61	swap	159
2.24.3.62	swap	159
2.24.3.63	value	160
2.25	Type Traits	161
2.25.1	Detailed Description	166
2.25.2	Typedef Documentation	166
2.25.2.1	false_type	166
2.25.2.2	true_type	166
2.26	Decimal Floating-Point Arithmetic	167

2.26.1 Detailed Description	167
2.27 Binder Classes	168
2.27.1 Detailed Description	169
2.27.2 Function Documentation	169
2.27.2.1 bind	169
2.27.2.2 bind1st	170
2.27.2.3 bind2nd	170
2.28 Algorithms	171
2.28.1 Detailed Description	171
2.29 Mutating Algorithms	172
2.29.1 Function Documentation	174
2.29.1.1 copy	174
2.29.1.2 copy_backward	175
2.29.1.3 copy_if	175
2.29.1.4 copy_n	176
2.29.1.5 fill	176
2.29.1.6 fill_n	177
2.29.1.7 generate	177
2.29.1.8 generate_n	178
2.29.1.9 is_partitioned	178
2.29.1.10 iter_swap	179
2.29.1.11 move	179
2.29.1.12 move	180
2.29.1.13 move_backward	180
2.29.1.14 partition	181
2.29.1.15 partition_copy	181
2.29.1.16 partition_point	182
2.29.1.17 random_shuffle	182
2.29.1.18 random_shuffle	183
2.29.1.19 remove	183
2.29.1.20 remove_copy	184

2.29.1.21	remove_copy_if	184
2.29.1.22	remove_if	185
2.29.1.23	replace	185
2.29.1.24	replace_copy_if	186
2.29.1.25	replace_if	186
2.29.1.26	reverse	187
2.29.1.27	reverse_copy	187
2.29.1.28	rotate	188
2.29.1.29	rotate_copy	189
2.29.1.30	stable_partition	189
2.29.1.31	swap	190
2.29.1.32	swap_ranges	190
2.29.1.33	transform	191
2.29.1.34	transform	191
2.29.1.35	unique	192
2.29.1.36	unique	192
2.29.1.37	unique_copy	193
2.29.1.38	unique_copy	194
2.30	Non-Mutating Algorithms	195
2.30.1	Function Documentation	197
2.30.1.1	adjacent_find	197
2.30.1.2	adjacent_find	197
2.30.1.3	all_of	197
2.30.1.4	any_of	198
2.30.1.5	count	198
2.30.1.6	count_if	199
2.30.1.7	equal	199
2.30.1.8	equal	200
2.30.1.9	find	200
2.30.1.10	find_end	201
2.30.1.11	find_end	201

2.30.1.12	find_first_of	202
2.30.1.13	find_first_of	203
2.30.1.14	find_if	203
2.30.1.15	find_if_not	204
2.30.1.16	for_each	204
2.30.1.17	mismatch	205
2.30.1.18	mismatch	205
2.30.1.19	none_of	206
2.30.1.20	search	206
2.30.1.21	search	207
2.30.1.22	search_n	208
2.30.1.23	search_n	208
2.31	Sorting Algorithms	210
2.31.1	Function Documentation	213
2.31.1.1	inplace_merge	213
2.31.1.2	inplace_merge	213
2.31.1.3	is_sorted	214
2.31.1.4	is_sorted	214
2.31.1.5	is_sorted_until	215
2.31.1.6	is_sorted_until	215
2.31.1.7	lexicographical_compare	216
2.31.1.8	lexicographical_compare	216
2.31.1.9	max	217
2.31.1.10	max	217
2.31.1.11	max_element	217
2.31.1.12	max_element	218
2.31.1.13	merge	218
2.31.1.14	merge	219
2.31.1.15	min	220
2.31.1.16	min	220
2.31.1.17	min_element	221

2.31.1.18	min_element	221
2.31.1.19	minmax	221
2.31.1.20	minmax	222
2.31.1.21	minmax_element	222
2.31.1.22	minmax_element	223
2.31.1.23	next_permutation	223
2.31.1.24	next_permutation	224
2.31.1.25	nth_element	224
2.31.1.26	nth_element	225
2.31.1.27	partial_sort	225
2.31.1.28	partial_sort	226
2.31.1.29	partial_sort_copy	226
2.31.1.30	partial_sort_copy	227
2.31.1.31	prev_permutation	228
2.31.1.32	prev_permutation	228
2.31.1.33	sort	229
2.31.1.34	sort	229
2.31.1.35	stable_sort	230
2.31.1.36	stable_sort	230
2.32	Set Operation Algorithms	232
2.32.1	Detailed Description	233
2.32.2	Function Documentation	233
2.32.2.1	includes	233
2.32.2.2	includes	234
2.32.2.3	set_difference	234
2.32.2.4	set_difference	235
2.32.2.5	set_intersection	235
2.32.2.6	set_intersection	236
2.32.2.7	set_symmetric_difference	237
2.32.2.8	set_symmetric_difference	237
2.32.2.9	set_union	238

2.32.2.10	<code>set_union</code>	239
2.33	Binary Search Algorithms	240
2.33.1	Detailed Description	240
2.33.2	Function Documentation	241
2.33.2.1	<code>binary_search</code>	241
2.33.2.2	<code>binary_search</code>	241
2.33.2.3	<code>equal_range</code>	242
2.33.2.4	<code>equal_range</code>	243
2.33.2.5	<code>lower_bound</code>	243
2.33.2.6	<code>lower_bound</code>	244
2.33.2.7	<code>upper_bound</code>	244
2.33.2.8	<code>upper_bound</code>	245
2.34	Allocators	246
2.34.1	Detailed Description	247
2.35	Atomics	248
2.35.1	Detailed Description	254
2.35.2	Define Documentation	254
2.35.2.1	<code>_GLIBCXX_ATOMIC_PROPERTY</code>	254
2.35.3	Typedef Documentation	254
2.35.3.1	<code>atomic_char</code>	254
2.35.3.2	<code>atomic_char16_t</code>	254
2.35.3.3	<code>atomic_char32_t</code>	254
2.35.3.4	<code>atomic_int</code>	254
2.35.3.5	<code>atomic_llong</code>	254
2.35.3.6	<code>atomic_long</code>	255
2.35.3.7	<code>atomic_schar</code>	255
2.35.3.8	<code>atomic_short</code>	255
2.35.3.9	<code>atomic_uchar</code>	255
2.35.3.10	<code>atomic_uint</code>	255
2.35.3.11	<code>atomic_ullong</code>	255
2.35.3.12	<code>atomic_ulong</code>	255

2.35.3.13	atomic_ushort	256
2.35.3.14	atomic_wchar_t	256
2.35.3.15	memory_order	256
2.35.4	Enumeration Type Documentation	256
2.35.4.1	memory_order	256
2.35.5	Function Documentation	256
2.35.5.1	kill_dependency	256
2.36	Hashes	257
2.36.1	Detailed Description	257
2.37	Locales	258
2.37.1	Detailed Description	260
2.38	Random Number Generation	261
2.38.1	Detailed Description	261
2.38.2	Function Documentation	261
2.38.2.1	generate_canonical	261
2.39	Function Objects	262
2.39.1	Detailed Description	263
2.39.2	Function Documentation	263
2.39.2.1	mem_fn	263
2.40	Arithmetic Classes	264
2.40.1	Detailed Description	264
2.41	Comparison Classes	265
2.41.1	Detailed Description	265
2.42	Boolean Operations Classes	266
2.42.1	Detailed Description	266
2.43	Negators	267
2.43.1	Detailed Description	267
2.43.2	Function Documentation	268
2.43.2.1	not1	268
2.43.2.2	not2	268
2.44	Adaptors for pointers to functions	269

2.44.1	Detailed Description	269
2.44.2	Function Documentation	270
2.44.2.1	ptr_fun	270
2.44.2.2	ptr_fun	270
2.45	Adaptors for pointers to members	271
2.45.1	Detailed Description	272
2.46	Heap Algorithms	273
2.46.1	Function Documentation	274
2.46.1.1	is_heap	274
2.46.1.2	is_heap	274
2.46.1.3	is_heap_until	275
2.46.1.4	is_heap_until	275
2.46.1.5	make_heap	276
2.46.1.6	make_heap	276
2.46.1.7	pop_heap	276
2.46.1.8	pop_heap	277
2.46.1.9	push_heap	277
2.46.1.10	push_heap	277
2.46.1.11	sort_heap	278
2.46.1.12	sort_heap	278
2.47	Iterators	279
2.47.1	Detailed Description	283
2.47.2	Function Documentation	283
2.47.2.1	__iterator_category	283
2.47.2.2	back_inserter	283
2.47.2.3	front_inserter	284
2.47.2.4	inserter	284
2.47.2.5	operator!=	284
2.47.2.6	operator==	285
2.47.2.7	operator==	285
2.48	Policy-Based Data Structures	286

2.48.1 Detailed Description	287
2.49 Random Number Generators	288
2.49.1 Detailed Description	289
2.49.2 Typedef Documentation	289
2.49.2.1 minstd_rand	289
2.49.2.2 minstd_rand0	289
2.49.2.3 mt19937	290
2.49.2.4 mt19937_64	290
2.49.3 Function Documentation	290
2.49.3.1 operator<<	290
2.50 Random Number Distributions	291
2.51 Uniform Distributions	292
2.51.1 Function Documentation	292
2.51.1.1 operator<<	292
2.51.1.2 operator<<	293
2.51.1.3 operator>>	293
2.51.1.4 operator>>	294
2.52 Normal Distributions	295
2.52.1 Function Documentation	296
2.52.1.1 operator<<	296
2.52.1.2 operator>>	296
2.53 Bernoulli Distributions	297
2.53.1 Function Documentation	298
2.53.1.1 operator<<	298
2.53.1.2 operator<<	298
2.53.1.3 operator>>	299
2.53.1.4 operator>>	299
2.54 Poisson Distributions	300
2.54.1 Function Documentation	301
2.54.1.1 operator<<	301
2.54.1.2 operator<<	301

2.54.1.3	operator<<	302
2.54.1.4	operator>>	302
2.54.1.5	operator>>	303
2.54.1.6	operator>>	303
2.55	Random Number Utilities	305
3	Directory Documentation	307
3.1	include/backward/ Directory Reference	307
3.2	include/x86_64-unknown-linux-gnu/bits/ Directory Reference	309
3.3	include/bits/ Directory Reference	310
3.4	include/debug/ Directory Reference	313
3.5	include/decimal/ Directory Reference	314
3.6	include/ext/pb_ds/detail/ Directory Reference	315
3.7	/mnt/share/src/gcc-trunk/libstdc++-v3/doc/ Directory Reference	316
3.8	/mnt/share/src/gcc-trunk/libstdc++-v3/doc/doxygen/ Directory Reference	317
3.9	include/ext/ Directory Reference	318
3.10	/mnt/share/src/gcc-trunk/ Directory Reference	320
3.11	include/profile/impl/ Directory Reference	321
3.12	include/ Directory Reference	322
3.13	/mnt/share/src/gcc-trunk/libstdc++-v3/ Directory Reference	325
3.14	/mnt/share/src/gcc-trunk/libstdc++-v3/libsupc++/ Directory Reference	326
3.15	include/parallel/ Directory Reference	327
3.16	include/ext/pb_ds/ Directory Reference	329
3.17	include/profile/ Directory Reference	330
3.18	/mnt/share/src/ Directory Reference	331
3.19	include/tr1/ Directory Reference	332
3.20	include/tr1_impl/ Directory Reference	333
3.21	include/x86_64-unknown-linux-gnu/ Directory Reference	334
4	Namespace Documentation	335
4.1	__gnu_cxx Namespace Reference	335

4.1.1	Detailed Description	357
4.1.2	Function Documentation	357
4.1.2.1	__static_pointer_cast	357
4.1.2.2	__static_pointer_cast	358
4.1.2.3	_Bit_scan_forward	358
4.1.2.4	operator!=	358
4.1.2.5	operator!=	358
4.1.2.6	operator!=	359
4.1.2.7	operator+	359
4.1.2.8	operator+	360
4.1.2.9	operator+	360
4.1.2.10	operator+	361
4.1.2.11	operator+	361
4.1.2.12	operator<	362
4.1.2.13	operator<	362
4.1.2.14	operator<	363
4.1.2.15	operator<=	363
4.1.2.16	operator<=	363
4.1.2.17	operator<=	364
4.1.2.18	operator==	364
4.1.2.19	operator==	365
4.1.2.20	operator==	365
4.1.2.21	operator==	366
4.1.2.22	operator>	366
4.1.2.23	operator>	366
4.1.2.24	operator>	367
4.1.2.25	operator>=	367
4.1.2.26	operator>=	368
4.1.2.27	operator>=	368
4.1.2.28	swap	369
4.2	__gnu_cxx::__detail Namespace Reference	370

4.2.1	Detailed Description	370
4.2.2	Function Documentation	371
4.2.2.1	__bit_allocate	371
4.2.2.2	__bit_free	371
4.2.2.3	__num_bitmaps	371
4.2.2.4	__num_blocks	371
4.3	__gnu_cxx::typelist Namespace Reference	372
4.3.1	Detailed Description	372
4.3.2	Function Documentation	372
4.3.2.1	apply_generator	372
4.4	__gnu_debug Namespace Reference	373
4.4.1	Detailed Description	379
4.4.2	Function Documentation	379
4.4.2.1	__check_dereferenceable	379
4.4.2.2	__check_dereferenceable	379
4.4.2.3	__check_dereferenceable	380
4.4.2.4	__check_singular	380
4.4.2.5	__check_singular	380
4.4.2.6	__check_singular_aux	380
4.4.2.7	__check_string	380
4.4.2.8	__check_string	380
4.4.2.9	__valid_range	381
4.4.2.10	__valid_range	381
4.4.2.11	__valid_range_aux	381
4.4.2.12	__valid_range_aux	381
4.4.2.13	__valid_range_aux2	382
4.4.2.14	__valid_range_aux2	382
4.5	__gnu_internal Namespace Reference	383
4.5.1	Detailed Description	383
4.6	__gnu_parallel Namespace Reference	384
4.6.1	Detailed Description	400

4.6.2	Typedef Documentation	400
4.6.2.1	_BinIndex	400
4.6.2.2	_CASable	400
4.6.2.3	_SequenceIndex	401
4.6.2.4	_ThreadIndex	401
4.6.3	Enumeration Type Documentation	401
4.6.3.1	_AlgorithmStrategy	401
4.6.3.2	_FindAlgorithm	401
4.6.3.3	_MultiwayMergeAlgorithm	401
4.6.3.4	_Parallelism	401
4.6.3.5	_PartialSumAlgorithm	402
4.6.3.6	_SortAlgorithm	402
4.6.3.7	_SplittingAlgorithm	402
4.6.4	Function Documentation	402
4.6.4.1	__calc_borders	402
4.6.4.2	__compare_and_swap	403
4.6.4.3	__compare_and_swap_32	403
4.6.4.4	__compare_and_swap_64	403
4.6.4.5	__decode2	404
4.6.4.6	__determine_samples	404
4.6.4.7	__encode2	405
4.6.4.8	__fetch_and_add	405
4.6.4.9	__fetch_and_add_32	406
4.6.4.10	__fetch_and_add_64	406
4.6.4.11	__find_template	406
4.6.4.12	__find_template	407
4.6.4.13	__find_template	408
4.6.4.14	__find_template	408
4.6.4.15	__for_each_template_random_access	409
4.6.4.16	__for_each_template_random_access_ed	410
4.6.4.17	__for_each_template_random_access_omp_loop	410

4.6.4.18	<code>__for_each_template_random_access_omp_loop_-static</code>	411
4.6.4.19	<code>__for_each_template_random_access_workstealing</code>	412
4.6.4.20	<code>__is_sorted</code>	412
4.6.4.21	<code>__median_of_three_iterators</code>	413
4.6.4.22	<code>__merge_advance</code>	413
4.6.4.23	<code>__merge_advance_movc</code>	414
4.6.4.24	<code>__merge_advance_usual</code>	415
4.6.4.25	<code>__parallel_merge_advance</code>	415
4.6.4.26	<code>__parallel_merge_advance</code>	416
4.6.4.27	<code>__parallel_nth_element</code>	417
4.6.4.28	<code>__parallel_partial_sort</code>	417
4.6.4.29	<code>__parallel_partial_sum</code>	417
4.6.4.30	<code>__parallel_partial_sum_basecase</code>	418
4.6.4.31	<code>__parallel_partial_sum_linear</code>	419
4.6.4.32	<code>__parallel_partition</code>	419
4.6.4.33	<code>__parallel_random_shuffle</code>	420
4.6.4.34	<code>__parallel_random_shuffle_drs</code>	420
4.6.4.35	<code>__parallel_random_shuffle_drs_pu</code>	421
4.6.4.36	<code>__parallel_sort</code>	421
4.6.4.37	<code>__parallel_sort</code>	422
4.6.4.38	<code>__parallel_sort</code>	423
4.6.4.39	<code>__parallel_sort</code>	423
4.6.4.40	<code>__parallel_sort</code>	424
4.6.4.41	<code>__parallel_sort</code>	424
4.6.4.42	<code>__parallel_sort</code>	425
4.6.4.43	<code>__parallel_sort_qs</code>	425
4.6.4.44	<code>__parallel_sort_qs_conquer</code>	426
4.6.4.45	<code>__parallel_sort_qs_divide</code>	426
4.6.4.46	<code>__parallel_sort_qsb</code>	427
4.6.4.47	<code>__parallel_unique_copy</code>	427

4.6.4.48	<code>__parallel_unique_copy</code>	428
4.6.4.49	<code>__qsb_conquer</code>	428
4.6.4.50	<code>__qsb_divide</code>	429
4.6.4.51	<code>__qsb_local_sort_with_helping</code>	429
4.6.4.52	<code>__random_number_pow2</code>	430
4.6.4.53	<code>__rd_log2</code>	430
4.6.4.54	<code>__round_up_to_pow2</code>	430
4.6.4.55	<code>__search_template</code>	431
4.6.4.56	<code>__sequential_multiway_merge</code>	431
4.6.4.57	<code>__sequential_random_shuffle</code>	432
4.6.4.58	<code>__shrink</code>	432
4.6.4.59	<code>__shrink_and_double</code>	433
4.6.4.60	<code>__yield</code>	433
4.6.4.61	<code>equally_split</code>	433
4.6.4.62	<code>equally_split_point</code>	434
4.6.4.63	<code>list_partition</code>	434
4.6.4.64	<code>max</code>	435
4.6.4.65	<code>min</code>	435
4.6.4.66	<code>multiseq_partition</code>	436
4.6.4.67	<code>multiseq_selection</code>	436
4.6.4.68	<code>multiway_merge</code>	437
4.6.4.69	<code>multiway_merge_3_variant</code>	439
4.6.4.70	<code>multiway_merge_4_variant</code>	440
4.6.4.71	<code>multiway_merge_exact_splitting</code>	440
4.6.4.72	<code>multiway_merge_loser_tree</code>	441
4.6.4.73	<code>multiway_merge_loser_tree_sentinel</code>	441
4.6.4.74	<code>multiway_merge_loser_tree_unguarded</code>	442
4.6.4.75	<code>multiway_merge_sampling_splitting</code>	443
4.6.4.76	<code>multiway_merge_sentinels</code>	443
4.6.4.77	<code>parallel_multiway_merge</code>	445
4.6.4.78	<code>parallel_sort_mwms</code>	446

4.6.4.79	parallel_sort_mwms_pu	446
4.6.5	Variable Documentation	447
4.6.5.1	_CASable_bits	447
4.6.5.2	_CASable_mask	447
4.7	__gnu_pbds Namespace Reference	448
4.7.1	Detailed Description	450
4.8	__gnu_profile Namespace Reference	451
4.8.1	Detailed Description	455
4.8.2	Function Documentation	456
4.8.2.1	__get__global_lock	456
4.8.2.2	__profcxx_init	456
4.8.2.3	__report	456
4.9	__gnu_sequential Namespace Reference	457
4.9.1	Detailed Description	457
4.10	abi Namespace Reference	458
4.10.1	Detailed Description	458
4.11	std Namespace Reference	459
4.11.1	Detailed Description	598
4.11.2	Typedef Documentation	598
4.11.2.1	new_handler	598
4.11.2.2	streamoff	599
4.11.2.3	streampos	599
4.11.2.4	streamsize	599
4.11.2.5	u16streampos	599
4.11.2.6	u32streampos	599
4.11.2.7	wstreampos	599
4.11.3	Enumeration Type Documentation	600
4.11.3.1	"@52	600
4.11.3.2	float_denorm_style	600
4.11.3.3	float_round_style	600
4.11.4	Function Documentation	601

4.11.4.1	<code>__final_insertion_sort</code>	601
4.11.4.2	<code>__final_insertion_sort</code>	601
4.11.4.3	<code>__find</code>	601
4.11.4.4	<code>__find</code>	601
4.11.4.5	<code>__find_if</code>	602
4.11.4.6	<code>__find_if</code>	602
4.11.4.7	<code>__find_if_not</code>	602
4.11.4.8	<code>__find_if_not</code>	602
4.11.4.9	<code>__gcd</code>	602
4.11.4.10	<code>__heap_select</code>	603
4.11.4.11	<code>__heap_select</code>	603
4.11.4.12	<code>__inplace_stable_partition</code>	603
4.11.4.13	<code>__inplace_stable_sort</code>	603
4.11.4.14	<code>__inplace_stable_sort</code>	604
4.11.4.15	<code>__insertion_sort</code>	604
4.11.4.16	<code>__insertion_sort</code>	604
4.11.4.17	<code>__introsort_loop</code>	604
4.11.4.18	<code>__introsort_loop</code>	605
4.11.4.19	<code>__invoke</code>	605
4.11.4.20	<code>__lg</code>	605
4.11.4.21	<code>__merge_adaptive</code>	605
4.11.4.22	<code>__merge_adaptive</code>	606
4.11.4.23	<code>__merge_backward</code>	606
4.11.4.24	<code>__merge_backward</code>	606
4.11.4.25	<code>__merge_without_buffer</code>	606
4.11.4.26	<code>__merge_without_buffer</code>	607
4.11.4.27	<code>__move_median_first</code>	607
4.11.4.28	<code>__move_median_first</code>	607
4.11.4.29	<code>__partition</code>	607
4.11.4.30	<code>__partition</code>	608
4.11.4.31	<code>__reverse</code>	608

4.11.4.32	<code>__reverse</code>	608
4.11.4.33	<code>__rotate</code>	608
4.11.4.34	<code>__rotate</code>	609
4.11.4.35	<code>__rotate</code>	609
4.11.4.36	<code>__rotate_adaptive</code>	609
4.11.4.37	<code>__search_n</code>	610
4.11.4.38	<code>__search_n</code>	610
4.11.4.39	<code>__search_n</code>	610
4.11.4.40	<code>__search_n</code>	610
4.11.4.41	<code>__stable_partition_adaptive</code>	611
4.11.4.42	<code>__unguarded_insertion_sort</code>	611
4.11.4.43	<code>__unguarded_insertion_sort</code>	611
4.11.4.44	<code>__unguarded_linear_insert</code>	611
4.11.4.45	<code>__unguarded_linear_insert</code>	612
4.11.4.46	<code>__unguarded_partition</code>	612
4.11.4.47	<code>__unguarded_partition</code>	612
4.11.4.48	<code>__unguarded_partition_pivot</code>	612
4.11.4.49	<code>__unguarded_partition_pivot</code>	613
4.11.4.50	<code>__unique_copy</code>	613
4.11.4.51	<code>__unique_copy</code>	613
4.11.4.52	<code>__unique_copy</code>	613
4.11.4.53	<code>__unique_copy</code>	614
4.11.4.54	<code>__unique_copy</code>	614
4.11.4.55	<code>__unique_copy</code>	614
4.11.4.56	<code>_Construct</code>	614
4.11.4.57	<code>_Destroy</code>	614
4.11.4.58	<code>_Destroy</code>	615
4.11.4.59	<code>accumulate</code>	615
4.11.4.60	<code>accumulate</code>	615
4.11.4.61	<code>adjacent_difference</code>	616
4.11.4.62	<code>adjacent_difference</code>	616

4.11.4.63	advance	617
4.11.4.64	bind	617
4.11.4.65	boolalpha	618
4.11.4.66	cref	618
4.11.4.67	cref	618
4.11.4.68	dec	618
4.11.4.69	distance	618
4.11.4.70	endl	619
4.11.4.71	ends	619
4.11.4.72	fixed	620
4.11.4.73	flush	620
4.11.4.74	forward	620
4.11.4.75	forward	620
4.11.4.76	get_money	620
4.11.4.77	get_temporary_buffer	621
4.11.4.78	getline	621
4.11.4.79	getline	622
4.11.4.80	getline	623
4.11.4.81	getline	623
4.11.4.82	has_facet	624
4.11.4.83	hex	624
4.11.4.84	inner_product	624
4.11.4.85	inner_product	625
4.11.4.86	internal	625
4.11.4.87	iota	626
4.11.4.88	isalnum	626
4.11.4.89	isalpha	626
4.11.4.90	isctrl	626
4.11.4.91	isdigit	626
4.11.4.92	isgraph	626
4.11.4.93	islower	627

4.11.4.94 isprint	627
4.11.4.95 ispunct	627
4.11.4.96 isspace	627
4.11.4.97 isupper	627
4.11.4.98 isxdigit	627
4.11.4.99 left	627
4.11.4.100nboolalpha	628
4.11.4.101nshowbase	628
4.11.4.102nshowpoint	628
4.11.4.103nshowpos	628
4.11.4.104noskipws	628
4.11.4.105nunitbuf	628
4.11.4.106nouppercase	629
4.11.4.107oct	629
4.11.4.108operator!=	629
4.11.4.109operator!=	629
4.11.4.110operator!=	629
4.11.4.111operator!=	630
4.11.4.112operator!=	630
4.11.4.113operator!=	630
4.11.4.114operator!=	630
4.11.4.115operator!=	630
4.11.4.116operator!=	631
4.11.4.117operator!=	631
4.11.4.118operator!=	631
4.11.4.119operator!=	631
4.11.4.120operator!=	631
4.11.4.121operator!=	632
4.11.4.122operator!=	632
4.11.4.123operator!=	632
4.11.4.124operator!=	633

4.11.4.125operator&	633
4.11.4.126operator+	633
4.11.4.127operator+	634
4.11.4.128operator+	634
4.11.4.129operator+	634
4.11.4.130operator+	635
4.11.4.131operator<	635
4.11.4.132operator<	636
4.11.4.133operator<	636
4.11.4.134operator<	637
4.11.4.135operator<	637
4.11.4.136operator<	637
4.11.4.137operator<	638
4.11.4.138operator<	638
4.11.4.139operator<	639
4.11.4.140operator<	639
4.11.4.141operator<	640
4.11.4.142operator<	640
4.11.4.143operator<	641
4.11.4.144operator<	641
4.11.4.145operator<<	641
4.11.4.146operator<<	642
4.11.4.147operator<<	642
4.11.4.148operator<<	643
4.11.4.149operator<<	643
4.11.4.150operator<<	644
4.11.4.151operator<<	644
4.11.4.152operator<<	645
4.11.4.153operator<<	645
4.11.4.154operator<<	646
4.11.4.155operator<<	646

4.11.4.156operator<<	647
4.11.4.157operator<<	647
4.11.4.158operator<<	648
4.11.4.159operator<=	648
4.11.4.160operator<=	649
4.11.4.161operator<=	649
4.11.4.162operator<=	649
4.11.4.163operator<=	649
4.11.4.164operator<=	649
4.11.4.165operator<=	650
4.11.4.166operator<=	650
4.11.4.167operator<=	650
4.11.4.168operator<=	650
4.11.4.169operator<=	650
4.11.4.170operator<=	651
4.11.4.171operator<=	651
4.11.4.172operator<=	651
4.11.4.173operator==	652
4.11.4.174operator==	652
4.11.4.175operator==	653
4.11.4.176operator==	653
4.11.4.177operator==	654
4.11.4.178operator==	654
4.11.4.179operator==	654
4.11.4.180operator==	655
4.11.4.181operator==	655
4.11.4.182operator==	656
4.11.4.183operator==	656
4.11.4.184operator==	656
4.11.4.185operator==	657
4.11.4.186operator==	657

4.11.4.187operator==	657
4.11.4.188operator==	658
4.11.4.189operator==	658
4.11.4.190operator==	658
4.11.4.191operator>	659
4.11.4.192operator>	659
4.11.4.193operator>	659
4.11.4.194operator>	659
4.11.4.195operator>	659
4.11.4.196operator>	660
4.11.4.197operator>	660
4.11.4.198operator>	660
4.11.4.199operator>	660
4.11.4.200operator>	660
4.11.4.201operator>	661
4.11.4.202operator>	661
4.11.4.203operator>	661
4.11.4.204operator>	662
4.11.4.205operator>=	662
4.11.4.206operator>=	662
4.11.4.207operator>=	662
4.11.4.208operator>=	663
4.11.4.209operator>=	663
4.11.4.210operator>=	663
4.11.4.211operator>=	663
4.11.4.212operator>=	663
4.11.4.213operator>=	664
4.11.4.214operator>=	664
4.11.4.215operator>=	664
4.11.4.216operator>=	664
4.11.4.217operator>=	665

4.11.4.218operator>=	665
4.11.4.219operator>>	665
4.11.4.220operator>>	666
4.11.4.221operator>>	666
4.11.4.222operator>>	667
4.11.4.223operator>>	668
4.11.4.224operator>>	669
4.11.4.225operator>>	669
4.11.4.226operator>>	670
4.11.4.227operator>>	671
4.11.4.228operator>>	672
4.11.4.229operator>>	673
4.11.4.230operator^	673
4.11.4.231operator	673
4.11.4.232partial_sum	674
4.11.4.233partial_sum	674
4.11.4.234put_money	675
4.11.4.235ref	675
4.11.4.236ref	675
4.11.4.237replace_copy	675
4.11.4.238resetiosflags	676
4.11.4.239return_temporary_buffer	676
4.11.4.240right	676
4.11.4.241scientific	677
4.11.4.242set_new_handler	677
4.11.4.243setbase	677
4.11.4.244setfill	677
4.11.4.245setiosflags	678
4.11.4.246setprecision	678
4.11.4.247setw	678
4.11.4.248showbase	678

4.11.4.24	showpoint	679
4.11.4.25	showpos	679
4.11.4.25	skipws	679
4.11.4.25	swap	679
4.11.4.25	swap	679
4.11.4.25	swap	680
4.11.4.25	swap	680
4.11.4.25	swap	680
4.11.4.25	swap	680
4.11.4.25	swap	680
4.11.4.25	swap	680
4.11.4.25	swap	681
4.11.4.26	swap	681
4.11.4.26	lswap	681
4.11.4.26	swap	681
4.11.4.26	tolower	682
4.11.4.26	toupper	682
4.11.4.26	uninitialized_copy	682
4.11.4.26	uninitialized_copy_n	683
4.11.4.26	uninitialized_fill	683
4.11.4.26	uninitialized_fill_n	683
4.11.4.26	unitbuf	684
4.11.4.27	uppercase	684
4.11.4.27	use_facet	684
4.11.4.27	ws	685
4.11.5	Variable Documentation	685
4.11.5.1	cerr	685
4.11.5.2	cin	685
4.11.5.3	clog	686
4.11.5.4	cout	686
4.11.5.5	wcerr	686
4.11.5.6	wcin	686

4.11.5.7	wclog	686
4.11.5.8	wcout	686
4.12	std::__debug Namespace Reference	687
4.12.1	Detailed Description	692
4.13	std::__detail Namespace Reference	693
4.13.1	Detailed Description	693
4.14	std::__parallel Namespace Reference	694
4.14.1	Detailed Description	719
4.15	std::__profile Namespace Reference	720
4.15.1	Detailed Description	726
4.16	std::chrono Namespace Reference	727
4.16.1	Detailed Description	729
4.16.2	Typedef Documentation	730
4.16.2.1	hours	730
4.16.2.2	microseconds	730
4.16.2.3	milliseconds	730
4.16.2.4	minutes	730
4.16.2.5	nanoseconds	730
4.16.2.6	seconds	730
4.16.3	Function Documentation	731
4.16.3.1	duration_cast	731
4.16.3.2	time_point_cast	731
4.17	std::decimal Namespace Reference	732
4.17.1	Detailed Description	742
4.17.2	Function Documentation	742
4.17.2.1	decimal32_to_long_long	742
4.18	std::placeholders Namespace Reference	743
4.18.1	Detailed Description	743
4.19	std::regex_constants Namespace Reference	744
4.19.1	Detailed Description	745
4.19.2	Typedef Documentation	745

4.19.2.1	match_flag_type	745
4.19.2.2	syntax_option_type	746
4.19.3	Enumeration Type Documentation	746
4.19.3.1	__match_flag	746
4.19.3.2	__syntax_option	746
4.19.3.3	error_type	746
4.19.4	Function Documentation	746
4.19.4.1	error_backref	746
4.19.4.2	error_badbrace	747
4.19.4.3	error_badrepeat	747
4.19.4.4	error_brace	747
4.19.4.5	error_brack	747
4.19.4.6	error_collate	747
4.19.4.7	error_complexity	747
4.19.4.8	error_ctype	747
4.19.4.9	error_escape	747
4.19.4.10	error_paren	748
4.19.4.11	error_range	748
4.19.4.12	error_space	748
4.19.4.13	error_stack	748
4.19.5	Variable Documentation	748
4.19.5.1	awk	748
4.19.5.2	basic	748
4.19.5.3	collate	749
4.19.5.4	ECMAScript	749
4.19.5.5	egrep	749
4.19.5.6	extended	749
4.19.5.7	format_default	749
4.19.5.8	format_first_only	750
4.19.5.9	format_no_copy	750
4.19.5.10	format_sed	750

4.19.5.11	grep	751
4.19.5.12	icase	751
4.19.5.13	match_any	751
4.19.5.14	match_continuous	751
4.19.5.15	match_default	751
4.19.5.16	match_not_bol	751
4.19.5.17	match_not_bow	752
4.19.5.18	match_not_eol	752
4.19.5.19	match_not_eow	752
4.19.5.20	match_not_null	752
4.19.5.21	match_prev_avail	752
4.19.5.22	nosubs	752
4.19.5.23	optimize	753
4.20	std::rel_ops Namespace Reference	754
4.20.1	Detailed Description	754
4.20.2	Function Documentation	754
4.20.2.1	operator!=	754
4.20.2.2	operator<=	754
4.20.2.3	operator>	755
4.20.2.4	operator>=	755
4.21	std::this_thread Namespace Reference	756
4.21.1	Detailed Description	756
4.21.2	Function Documentation	756
4.21.2.1	get_id	756
4.21.2.2	sleep_for	756
4.21.2.3	sleep_until	756
4.21.2.4	yield	757
4.22	std::tr1 Namespace Reference	758
4.22.1	Detailed Description	761
4.23	std::tr1::__detail Namespace Reference	762
4.23.1	Detailed Description	762

5	Class Documentation	763
5.1	<code>__atomic0::atomic_address</code> Struct Reference	763
5.1.1	Detailed Description	764
5.2	<code>__atomic0::atomic_bool</code> Struct Reference	765
5.2.1	Detailed Description	765
5.3	<code>__atomic0::atomic_flag</code> Struct Reference	766
5.3.1	Detailed Description	766
5.4	<code>__atomic2::atomic_address</code> Struct Reference	767
5.4.1	Detailed Description	767
5.5	<code>__atomic2::atomic_bool</code> Struct Reference	768
5.5.1	Detailed Description	768
5.6	<code>__atomic2::atomic_flag</code> Struct Reference	769
5.6.1	Detailed Description	769
5.7	<code>__cxxabiv1::__forced_unwind</code> Class Reference	770
5.7.1	Detailed Description	770
5.8	<code>__gnu_cxx::__common_pool_policy< _PoolTp, _Thread ></code> Struct Template Reference	771
5.8.1	Detailed Description	771
5.9	<code>__gnu_cxx::__detail::__mini_vector< _Tp ></code> Class Template Reference	772
5.9.1	Detailed Description	772
5.10	<code>__gnu_cxx::__detail::_Bitmap_counter< _Tp ></code> Class Template Ref- erence	774
5.10.1	Detailed Description	774
5.11	<code>__gnu_cxx::__detail::_Ffit_finder< _Tp ></code> Class Template Reference	775
5.11.1	Detailed Description	775
5.11.2	Member Typedef Documentation	776
5.11.2.1	<code>argument_type</code>	776
5.11.2.2	<code>result_type</code>	776
5.12	<code>__gnu_cxx::__mt_alloc< _Tp, _Poolp ></code> Class Template Reference .	777
5.12.1	Detailed Description	778
5.13	<code>__gnu_cxx::__mt_alloc_base< _Tp ></code> Class Template Reference . . .	779
5.13.1	Detailed Description	779

5.14	<code>__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread ></code> Struct Template Reference	781
5.14.1	Detailed Description	781
5.15	<code>__gnu_cxx::__pool< false ></code> Class Template Reference	782
5.15.1	Detailed Description	783
5.16	<code>__gnu_cxx::__pool< true ></code> Class Template Reference	784
5.16.1	Detailed Description	785
5.17	<code>__gnu_cxx::__pool_alloc< _Tp ></code> Class Template Reference	786
5.17.1	Detailed Description	787
5.18	<code>__gnu_cxx::__pool_alloc_base</code> Class Reference	788
5.18.1	Detailed Description	788
5.19	<code>__gnu_cxx::__pool_base</code> Struct Reference	790
5.19.1	Detailed Description	790
5.20	<code>__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc ></code> Class Tem- plate Reference	791
5.20.1	Detailed Description	793
5.21	<code>__gnu_cxx::__scoped_lock</code> Class Reference	794
5.21.1	Detailed Description	794
5.22	<code>__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base ></code> Class Template Reference	795
5.22.1	Detailed Description	799
5.22.2	Constructor & Destructor Documentation	800
5.22.2.1	<code>__versa_string</code>	800
5.22.2.2	<code>__versa_string</code>	800
5.22.2.3	<code>__versa_string</code>	800
5.22.2.4	<code>__versa_string</code>	800
5.22.2.5	<code>__versa_string</code>	801
5.22.2.6	<code>__versa_string</code>	801
5.22.2.7	<code>__versa_string</code>	801
5.22.2.8	<code>__versa_string</code>	802
5.22.2.9	<code>__versa_string</code>	802
5.22.2.10	<code>__versa_string</code>	803

5.22.2.11	<code>__versa_string</code>	803
5.22.2.12	<code>~__versa_string</code>	803
5.22.3	Member Function Documentation	804
5.22.3.1	<code>append</code>	804
5.22.3.2	<code>append</code>	804
5.22.3.3	<code>append</code>	805
5.22.3.4	<code>append</code>	805
5.22.3.5	<code>append</code>	805
5.22.3.6	<code>append</code>	806
5.22.3.7	<code>append</code>	806
5.22.3.8	<code>assign</code>	807
5.22.3.9	<code>assign</code>	807
5.22.3.10	<code>assign</code>	808
5.22.3.11	<code>assign</code>	808
5.22.3.12	<code>assign</code>	809
5.22.3.13	<code>assign</code>	809
5.22.3.14	<code>assign</code>	810
5.22.3.15	<code>assign</code>	810
5.22.3.16	<code>at</code>	810
5.22.3.17	<code>at</code>	811
5.22.3.18	<code>back</code>	811
5.22.3.19	<code>back</code>	812
5.22.3.20	<code>begin</code>	812
5.22.3.21	<code>begin</code>	812
5.22.3.22	<code>c_str</code>	812
5.22.3.23	<code>capacity</code>	812
5.22.3.24	<code>cbegin</code>	813
5.22.3.25	<code>cend</code>	813
5.22.3.26	<code>clear</code>	813
5.22.3.27	<code>compare</code>	813
5.22.3.28	<code>compare</code>	814

5.22.3.29 compare	815
5.22.3.30 compare	815
5.22.3.31 compare	816
5.22.3.32 compare	817
5.22.3.33 copy	817
5.22.3.34 crbegin	818
5.22.3.35 crend	818
5.22.3.36 data	818
5.22.3.37 empty	819
5.22.3.38 end	819
5.22.3.39 end	819
5.22.3.40 erase	819
5.22.3.41 erase	820
5.22.3.42 erase	820
5.22.3.43 find	821
5.22.3.44 find	821
5.22.3.45 find	822
5.22.3.46 find	822
5.22.3.47 find_first_not_of	823
5.22.3.48 find_first_not_of	824
5.22.3.49 find_first_not_of	824
5.22.3.50 find_first_not_of	825
5.22.3.51 find_first_of	825
5.22.3.52 find_first_of	826
5.22.3.53 find_first_of	826
5.22.3.54 find_first_of	827
5.22.3.55 find_last_not_of	827
5.22.3.56 find_last_not_of	828
5.22.3.57 find_last_not_of	828
5.22.3.58 find_last_not_of	829
5.22.3.59 find_last_of	829

5.22.3.60	find_last_of	830
5.22.3.61	find_last_of	830
5.22.3.62	find_last_of	831
5.22.3.63	front	832
5.22.3.64	front	832
5.22.3.65	get_allocator	832
5.22.3.66	insert	832
5.22.3.67	insert	833
5.22.3.68	insert	833
5.22.3.69	insert	834
5.22.3.70	insert	835
5.22.3.71	insert	835
5.22.3.72	insert	836
5.22.3.73	insert	836
5.22.3.74	insert	837
5.22.3.75	length	837
5.22.3.76	max_size	838
5.22.3.77	operator+=	838
5.22.3.78	operator+=	838
5.22.3.79	operator+=	839
5.22.3.80	operator+=	839
5.22.3.81	operator=	839
5.22.3.82	operator=	840
5.22.3.83	operator=	840
5.22.3.84	operator=	840
5.22.3.85	operator=	841
5.22.3.86	operator[]	841
5.22.3.87	operator[]	841
5.22.3.88	push_back	842
5.22.3.89	rbegin	842
5.22.3.90	rbegin	842

5.22.3.91	rend	843
5.22.3.92	rend	843
5.22.3.93	replace	843
5.22.3.94	replace	844
5.22.3.95	replace	844
5.22.3.96	replace	845
5.22.3.97	replace	846
5.22.3.98	replace	846
5.22.3.99	replace	847
5.22.3.100	replace	848
5.22.3.101	replace	848
5.22.3.102	replace	849
5.22.3.103	replace	850
5.22.3.104	reserve	850
5.22.3.105	resize	851
5.22.3.106	resize	851
5.22.3.107	rfind	852
5.22.3.108	rfind	852
5.22.3.109	rfind	853
5.22.3.110	rfind	853
5.22.3.111	lshrink_to_fit	854
5.22.3.112	size	854
5.22.3.113	substr	855
5.22.3.114	swap	855
5.22.4	Member Data Documentation	856
5.22.4.1	npos	856
5.23	__gnu_cxx::_Caster<_ToType> Struct Template Reference	857
5.23.1	Detailed Description	857
5.24	__gnu_cxx::_Char_types<_CharT> Struct Template Reference	858
5.24.1	Detailed Description	858
5.25	__gnu_cxx::_ExtPtr_allocator<_Tp> Class Template Reference	859

5.25.1 Detailed Description	860
5.26 <code>__gnu_cxx::_Invalid_type</code> Struct Reference	861
5.26.1 Detailed Description	861
5.27 <code>__gnu_cxx::_Pointer_adapter< _Storage_policy ></code> Class Template Reference	862
5.27.1 Detailed Description	864
5.28 <code>__gnu_cxx::_Relative_pointer_impl< _Tp ></code> Class Template Reference	865
5.28.1 Detailed Description	865
5.29 <code>__gnu_cxx::_Relative_pointer_impl< const _Tp ></code> Class Template Reference	866
5.29.1 Detailed Description	866
5.30 <code>__gnu_cxx::_Std_pointer_impl< _Tp ></code> Class Template Reference . .	867
5.30.1 Detailed Description	867
5.31 <code>__gnu_cxx::_Unqualified_type< _Tp ></code> Struct Template Reference .	868
5.31.1 Detailed Description	868
5.32 <code>__gnu_cxx::annotate_base</code> Struct Reference	869
5.32.1 Detailed Description	869
5.33 <code>__gnu_cxx::array_allocator< _Tp, _Array ></code> Class Template Reference	870
5.33.1 Detailed Description	871
5.34 <code>__gnu_cxx::array_allocator_base< _Tp ></code> Class Template Reference .	872
5.34.1 Detailed Description	872
5.35 <code>__gnu_cxx::binary_compose< _Operation1, _Operation2, _-</code> <code>Operation3 ></code> Class Template Reference	874
5.35.1 Detailed Description	874
5.35.2 Member Typedef Documentation	875
5.35.2.1 <code>argument_type</code>	875
5.35.2.2 <code>result_type</code>	875
5.36 <code>__gnu_cxx::bitmap_allocator< _Tp ></code> Class Template Reference . . .	876
5.36.1 Detailed Description	877
5.36.2 Member Function Documentation	877
5.36.2.1 <code>_M_allocate_single_object</code>	877
5.36.2.2 <code>_M_deallocate_single_object</code>	878

5.37	<code>__gnu_cxx::char_traits<_CharT></code> Struct Template Reference	879
5.37.1	Detailed Description	880
5.38	<code>__gnu_cxx::character<V, I, S></code> Struct Template Reference	881
5.38.1	Detailed Description	881
5.39	<code>__gnu_cxx::condition_base</code> Struct Reference	882
5.39.1	Detailed Description	882
5.40	<code>__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2></code> Struct Template Reference	883
5.40.1	Detailed Description	883
5.41	<code>__gnu_cxx::constant_unary_fun<_Result, _Argument></code> Struct Template Reference	884
5.41.1	Detailed Description	884
5.42	<code>__gnu_cxx::constant_void_fun<_Result></code> Struct Template Reference	885
5.42.1	Detailed Description	885
5.43	<code>__gnu_cxx::debug_allocator<_Alloc></code> Class Template Reference	886
5.43.1	Detailed Description	886
5.44	<code>__gnu_cxx::enc_filebuf<_CharT></code> Class Template Reference	887
5.44.1	Detailed Description	890
5.44.2	Member Typedef Documentation	890
5.44.2.1	<code>__streambuf_type</code>	890
5.44.2.2	<code>char_type</code>	890
5.44.2.3	<code>int_type</code>	891
5.44.2.4	<code>off_type</code>	891
5.44.2.5	<code>pos_type</code>	891
5.44.2.6	<code>traits_type</code>	891
5.44.3	Member Function Documentation	892
5.44.3.1	<code>_M_create_pback</code>	892
5.44.3.2	<code>_M_destroy_pback</code>	892
5.44.3.3	<code>_M_set_buffer</code>	892
5.44.3.4	<code>close</code>	892
5.44.3.5	<code>eback</code>	893
5.44.3.6	<code>egptr</code>	893

5.44.3.7	epptr	893
5.44.3.8	gbump	894
5.44.3.9	getloc	894
5.44.3.10	gptr	894
5.44.3.11	imbue	894
5.44.3.12	in_avail	895
5.44.3.13	is_open	895
5.44.3.14	open	895
5.44.3.15	open	896
5.44.3.16	overflow	896
5.44.3.17	pbackfail	897
5.44.3.18	pbase	897
5.44.3.19	pbump	898
5.44.3.20	pptr	898
5.44.3.21	pubimbue	898
5.44.3.22	pubseekoff	899
5.44.3.23	pubseekpos	899
5.44.3.24	pubsetbuf	899
5.44.3.25	pubsync	900
5.44.3.26	sbumpc	900
5.44.3.27	seekoff	900
5.44.3.28	seekpos	901
5.44.3.29	setbuf	901
5.44.3.30	setg	901
5.44.3.31	setp	902
5.44.3.32	sgetc	902
5.44.3.33	sgetn	902
5.44.3.34	showmanyc	903
5.44.3.35	snextc	903
5.44.3.36	sputbackc	904
5.44.3.37	sputc	904

5.44.3.38	sputn	904
5.44.3.39	stossc	905
5.44.3.40	sungetc	905
5.44.3.41	sync	905
5.44.3.42	uflow	906
5.44.3.43	underflow	906
5.44.3.44	xsgetn	907
5.44.3.45	xspun	907
5.44.4	Member Data Documentation	908
5.44.4.1	_M_buf	908
5.44.4.2	_M_buf_locale	908
5.44.4.3	_M_buf_size	908
5.44.4.4	_M_ext_buf	908
5.44.4.5	_M_ext_buf_size	908
5.44.4.6	_M_ext_next	909
5.44.4.7	_M_in_beg	909
5.44.4.8	_M_in_cur	909
5.44.4.9	_M_in_end	909
5.44.4.10	_M_mode	910
5.44.4.11	_M_out_beg	910
5.44.4.12	_M_out_cur	910
5.44.4.13	_M_out_end	910
5.44.4.14	_M_pback	911
5.44.4.15	_M_pback_cur_save	911
5.44.4.16	_M_pback_end_save	911
5.44.4.17	_M_pback_init	911
5.44.4.18	_M_reading	912
5.45	<code>__gnu_cxx::encoding_char_traits<_CharT></code> Struct Template Reference	913
5.45.1	Detailed Description	914
5.46	<code>__gnu_cxx::encoding_state</code> Class Reference	915
5.46.1	Detailed Description	916

5.47	<code>__gnu_cxx::forced_error</code> Struct Reference	917
5.47.1	Detailed Description	917
5.47.2	Member Function Documentation	917
5.47.2.1	what	917
5.48	<code>__gnu_cxx::free_list</code> Class Reference	918
5.48.1	Detailed Description	918
5.48.2	Member Function Documentation	918
5.48.2.1	<code>_M_clear</code>	918
5.48.2.2	<code>_M_get</code>	919
5.48.2.3	<code>_M_insert</code>	919
5.49	<code>__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	920
5.49.1	Detailed Description	921
5.50	<code>__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _-</code> <code>Alloc ></code> Class Template Reference	923
5.50.1	Detailed Description	924
5.51	<code>__gnu_cxx::hash_multiset<_Value, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	926
5.51.1	Detailed Description	927
5.52	<code>__gnu_cxx::hash_set<_Value, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	928
5.52.1	Detailed Description	929
5.53	<code>__gnu_cxx::limit_condition</code> Struct Reference	930
5.53.1	Detailed Description	930
5.54	<code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference	931
5.54.1	Detailed Description	931
5.55	<code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference	932
5.55.1	Detailed Description	932
5.56	<code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference	933
5.56.1	Detailed Description	933
5.57	<code>__gnu_cxx::malloc_allocator<_Tp ></code> Class Template Reference	934
5.57.1	Detailed Description	934

5.58	<code>__gnu_cxx::new_allocator<_Tp></code> Class Template Reference	936
5.58.1	Detailed Description	937
5.59	<code>__gnu_cxx::project1st<_Arg1, _Arg2></code> Struct Template Reference	938
5.59.1	Detailed Description	938
5.59.2	Member Typedef Documentation	938
5.59.2.1	<code>first_argument_type</code>	938
5.59.2.2	<code>result_type</code>	938
5.59.2.3	<code>second_argument_type</code>	939
5.60	<code>__gnu_cxx::project2nd<_Arg1, _Arg2></code> Struct Template Reference	940
5.60.1	Detailed Description	940
5.60.2	Member Typedef Documentation	940
5.60.2.1	<code>first_argument_type</code>	940
5.60.2.2	<code>result_type</code>	940
5.60.2.3	<code>second_argument_type</code>	941
5.61	<code>__gnu_cxx::random_condition</code> Struct Reference	942
5.61.1	Detailed Description	942
5.62	<code>__gnu_cxx::random_condition::always_adjustor</code> Struct Reference	943
5.62.1	Detailed Description	943
5.63	<code>__gnu_cxx::random_condition::group_adjustor</code> Struct Reference	944
5.63.1	Detailed Description	944
5.64	<code>__gnu_cxx::random_condition::never_adjustor</code> Struct Reference	945
5.64.1	Detailed Description	945
5.65	<code>__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc></code> Struct Template Reference	946
5.65.1	Detailed Description	949
5.66	<code>__gnu_cxx::recursive_init_error</code> Class Reference	950
5.66.1	Detailed Description	950
5.66.2	Member Function Documentation	950
5.66.2.1	<code>what</code>	950
5.67	<code>__gnu_cxx::rope<_CharT, _Alloc></code> Class Template Reference	952
5.67.1	Detailed Description	958

5.68	__gnu_cxx::select1st<_Pair> Struct Template Reference	959
5.68.1	Detailed Description	959
5.68.2	Member Typedef Documentation	959
5.68.2.1	argument_type	959
5.68.2.2	result_type	959
5.69	__gnu_cxx::select2nd<_Pair> Struct Template Reference	960
5.69.1	Detailed Description	960
5.69.2	Member Typedef Documentation	960
5.69.2.1	argument_type	960
5.69.2.2	result_type	960
5.70	__gnu_cxx::slist<_Tp, _Alloc> Class Template Reference	961
5.70.1	Detailed Description	963
5.71	__gnu_cxx::stdio_filebuf<_CharT, _Traits> Class Template Reference	964
5.71.1	Detailed Description	968
5.71.2	Member Typedef Documentation	968
5.71.2.1	__streambuf_type	968
5.71.2.2	char_type	968
5.71.2.3	int_type	968
5.71.2.4	off_type	969
5.71.2.5	pos_type	969
5.71.2.6	traits_type	969
5.71.3	Constructor & Destructor Documentation	969
5.71.3.1	stdio_filebuf	969
5.71.3.2	stdio_filebuf	970
5.71.3.3	stdio_filebuf	970
5.71.3.4	~stdio_filebuf	971
5.71.4	Member Function Documentation	971
5.71.4.1	_M_create_pback	971
5.71.4.2	_M_destroy_pback	971
5.71.4.3	_M_set_buffer	971
5.71.4.4	close	972

5.71.4.5	eback	972
5.71.4.6	egptr	973
5.71.4.7	epptr	973
5.71.4.8	fd	973
5.71.4.9	file	974
5.71.4.10	gbump	974
5.71.4.11	getloc	974
5.71.4.12	gptr	975
5.71.4.13	imbue	975
5.71.4.14	in_avail	976
5.71.4.15	is_open	976
5.71.4.16	open	976
5.71.4.17	open	977
5.71.4.18	overflow	978
5.71.4.19	pbackfail	978
5.71.4.20	pbase	979
5.71.4.21	pbump	980
5.71.4.22	pptr	980
5.71.4.23	pubimbue	980
5.71.4.24	pubseekoff	981
5.71.4.25	pubseekpos	981
5.71.4.26	pubsetbuf	981
5.71.4.27	pubsync	982
5.71.4.28	sbumpc	982
5.71.4.29	seekoff	982
5.71.4.30	seekpos	983
5.71.4.31	setbuf	983
5.71.4.32	setg	984
5.71.4.33	setp	984
5.71.4.34	sgetc	985
5.71.4.35	sgetn	985

5.71.4.36	showmanyc	985
5.71.4.37	snextc	986
5.71.4.38	sputbackc	986
5.71.4.39	sputc	987
5.71.4.40	sputn	987
5.71.4.41	stosscc	988
5.71.4.42	sungetc	988
5.71.4.43	sync	988
5.71.4.44	uflow	989
5.71.4.45	underflow	989
5.71.4.46	xsgetn	990
5.71.4.47	xspu ^t n	991
5.71.5	Member Data Documentation	992
5.71.5.1	_M_buf	992
5.71.5.2	_M_buf_locale	992
5.71.5.3	_M_buf_size	992
5.71.5.4	_M_ext_buf	992
5.71.5.5	_M_ext_buf_size	993
5.71.5.6	_M_ext_next	993
5.71.5.7	_M_in_beg	993
5.71.5.8	_M_in_cur	993
5.71.5.9	_M_in_end	994
5.71.5.10	_M_mode	994
5.71.5.11	_M_out_beg	994
5.71.5.12	_M_out_cur	995
5.71.5.13	_M_out_end	995
5.71.5.14	_M_pback	995
5.71.5.15	_M_pback_cur_save	995
5.71.5.16	_M_pback_end_save	996
5.71.5.17	_M_pback_init	996
5.71.5.18	_M_reading	996

5.72 <code>__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits ></code> Class Template Reference	998
5.72.1 Detailed Description	1001
5.72.2 Member Typedef Documentation	1001
5.72.2.1 <code>__streambuf_type</code>	1001
5.72.2.2 <code>char_type</code>	1002
5.72.2.3 <code>int_type</code>	1002
5.72.2.4 <code>off_type</code>	1002
5.72.2.5 <code>pos_type</code>	1002
5.72.2.6 <code>traits_type</code>	1002
5.72.3 Member Function Documentation	1003
5.72.3.1 <code>eback</code>	1003
5.72.3.2 <code>egptr</code>	1003
5.72.3.3 <code>epptr</code>	1004
5.72.3.4 <code>file</code>	1004
5.72.3.5 <code>gbump</code>	1004
5.72.3.6 <code>getloc</code>	1005
5.72.3.7 <code>gptr</code>	1005
5.72.3.8 <code>imbue</code>	1005
5.72.3.9 <code>in_avail</code>	1006
5.72.3.10 <code>overflow</code>	1006
5.72.3.11 <code>pbackfail</code>	1007
5.72.3.12 <code>pbase</code>	1008
5.72.3.13 <code>pbump</code>	1008
5.72.3.14 <code>pptr</code>	1008
5.72.3.15 <code>pubimbue</code>	1009
5.72.3.16 <code>pubseekoff</code>	1009
5.72.3.17 <code>pubseekpos</code>	1010
5.72.3.18 <code>pubsetbuf</code>	1010
5.72.3.19 <code>pubsync</code>	1010
5.72.3.20 <code>sbumpc</code>	1010

5.72.3.21	<code>seekoff</code>	1011
5.72.3.22	<code>seekpos</code>	1011
5.72.3.23	<code>setbuf</code>	1012
5.72.3.24	<code>setg</code>	1012
5.72.3.25	<code>setp</code>	1012
5.72.3.26	<code>sgetc</code>	1013
5.72.3.27	<code>sgetn</code>	1013
5.72.3.28	<code>showmanyc</code>	1014
5.72.3.29	<code>snextc</code>	1014
5.72.3.30	<code>sputbackc</code>	1015
5.72.3.31	<code>sputc</code>	1015
5.72.3.32	<code>sputn</code>	1016
5.72.3.33	<code>stossc</code>	1016
5.72.3.34	<code>sungetc</code>	1016
5.72.3.35	<code>sync</code>	1017
5.72.3.36	<code>uflow</code>	1017
5.72.3.37	<code>underflow</code>	1017
5.72.3.38	<code>xsgetn</code>	1018
5.72.3.39	<code>xsputn</code>	1019
5.72.4	Member Data Documentation	1019
5.72.4.1	<code>_M_buf_locale</code>	1019
5.72.4.2	<code>_M_in_beg</code>	1019
5.72.4.3	<code>_M_in_cur</code>	1020
5.72.4.4	<code>_M_in_end</code>	1020
5.72.4.5	<code>_M_out_beg</code>	1020
5.72.4.6	<code>_M_out_cur</code>	1021
5.72.4.7	<code>_M_out_end</code>	1021
5.73	<code>__gnu_cxx::subtractive_rng</code> Class Reference	1022
5.73.1	Detailed Description	1022
5.73.2	Member Typedef Documentation	1022
5.73.2.1	<code>argument_type</code>	1022

5.73.2.2	result_type	1023
5.73.3	Constructor & Destructor Documentation	1023
5.73.3.1	subtractive_rng	1023
5.73.3.2	subtractive_rng	1023
5.73.4	Member Function Documentation	1023
5.73.4.1	operator()	1023
5.74	<code>__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp></code> Struct Template Reference	1024
5.74.1	Detailed Description	1025
5.74.2	Constructor & Destructor Documentation	1025
5.74.2.1	temporary_buffer	1025
5.74.2.2	~temporary_buffer	1025
5.74.3	Member Function Documentation	1026
5.74.3.1	begin	1026
5.74.3.2	end	1026
5.74.3.3	requested_size	1026
5.74.3.4	size	1026
5.75	<code>__gnu_cxx::throw_allocator_base<_Tp, _Cond></code> Class Template Reference	1027
5.75.1	Detailed Description	1028
5.76	<code>__gnu_cxx::throw_allocator_limit<_Tp></code> Struct Template Reference	1029
5.76.1	Detailed Description	1030
5.77	<code>__gnu_cxx::throw_allocator_random<_Tp></code> Struct Template Reference	1031
5.77.1	Detailed Description	1032
5.78	<code>__gnu_cxx::throw_value_base<_Cond></code> Struct Template Reference	1033
5.78.1	Detailed Description	1033
5.79	<code>__gnu_cxx::throw_value_limit</code> Struct Reference	1034
5.79.1	Detailed Description	1035
5.80	<code>__gnu_cxx::throw_value_random</code> Struct Reference	1036
5.80.1	Detailed Description	1037
5.81	<code>__gnu_cxx::unary_compose<_Operation1, _Operation2></code> Class Template Reference	1038

5.81.1	Detailed Description	1038
5.81.2	Member Typedef Documentation	1039
5.81.2.1	argument_type	1039
5.81.2.2	result_type	1039
5.82	<code>__gnu_debug::__is_same<_Type1, _Type2 ></code> Struct Template Reference	1040
5.82.1	Detailed Description	1040
5.83	<code>__gnu_debug::__After_nth_from<_Iterator ></code> Class Template Reference	1041
5.83.1	Detailed Description	1041
5.84	<code>__gnu_debug::__Not_equal_to<_Type ></code> Class Template Reference	1042
5.84.1	Detailed Description	1042
5.85	<code>__gnu_debug::__Safe_iterator<_Iterator, _Sequence ></code> Class Template Reference	1043
5.85.1	Detailed Description	1045
5.85.2	Constructor & Destructor Documentation	1045
5.85.2.1	<code>_Safe_iterator</code>	1045
5.85.2.2	<code>_Safe_iterator</code>	1045
5.85.2.3	<code>_Safe_iterator</code>	1046
5.85.2.4	<code>_Safe_iterator</code>	1046
5.85.3	Member Function Documentation	1046
5.85.3.1	<code>__attribute__</code>	1046
5.85.3.2	<code>__attribute__</code>	1047
5.85.3.3	<code>_M_attach</code>	1047
5.85.3.4	<code>_M_attach</code>	1047
5.85.3.5	<code>_M_attach_single</code>	1047
5.85.3.6	<code>_M_attach_single</code>	1047
5.85.3.7	<code>_M_attached_to</code>	1047
5.85.3.8	<code>_M_dereferenceable</code>	1048
5.85.3.9	<code>_M_detach</code>	1048
5.85.3.10	<code>_M_detach_single</code>	1048
5.85.3.11	<code>_M_get_distance</code>	1048
5.85.3.12	<code>_M_get_mutex</code>	1048

5.85.3.13	<code>_M_incrementable</code>	1049
5.85.3.14	<code>_M_invalidate</code>	1049
5.85.3.15	<code>_M_invalidate_single</code>	1049
5.85.3.16	<code>_M_is_begin</code>	1050
5.85.3.17	<code>_M_is_end</code>	1050
5.85.3.18	<code>base</code>	1050
5.85.3.19	<code>operator_iterator</code>	1050
5.85.3.20	<code>operator*</code>	1051
5.85.3.21	<code>operator++</code>	1051
5.85.3.22	<code>operator++</code>	1051
5.85.3.23	<code>operator--</code>	1052
5.85.3.24	<code>operator--</code>	1052
5.85.3.25	<code>operator-></code>	1052
5.85.3.26	<code>operator=</code>	1053
5.85.4	Member Data Documentation	1053
5.85.4.1	<code>_M_next</code>	1053
5.85.4.2	<code>_M_prior</code>	1053
5.85.4.3	<code>_M_sequence</code>	1053
5.85.4.4	<code>_M_version</code>	1054
5.86	<code>__gnu_debug::_Safe_iterator_base</code> Class Reference	1055
5.86.1	Detailed Description	1056
5.86.2	Constructor & Destructor Documentation	1056
5.86.2.1	<code>_Safe_iterator_base</code>	1056
5.86.2.2	<code>_Safe_iterator_base</code>	1056
5.86.2.3	<code>_Safe_iterator_base</code>	1056
5.86.3	Member Function Documentation	1057
5.86.3.1	<code>__attribute__</code>	1057
5.86.3.2	<code>__attribute__</code>	1057
5.86.3.3	<code>_M_attach</code>	1057
5.86.3.4	<code>_M_attach_single</code>	1057
5.86.3.5	<code>_M_attached_to</code>	1057

5.86.3.6	<code>_M_detach</code>	1057
5.86.3.7	<code>_M_detach_single</code>	1057
5.86.3.8	<code>_M_get_mutex</code>	1058
5.86.4	Member Data Documentation	1058
5.86.4.1	<code>_M_next</code>	1058
5.86.4.2	<code>_M_prior</code>	1058
5.86.4.3	<code>_M_sequence</code>	1058
5.86.4.4	<code>_M_version</code>	1058
5.87	<code>__gnu_debug::_Safe_sequence<_Sequence></code> Class Template Reference	1060
5.87.1	Detailed Description	1061
5.87.2	Member Function Documentation	1061
5.87.2.1	<code>_M_detach_all</code>	1061
5.87.2.2	<code>_M_detach_singular</code>	1061
5.87.2.3	<code>_M_get_mutex</code>	1061
5.87.2.4	<code>_M_invalidate_all</code>	1062
5.87.2.5	<code>_M_invalidate_if</code>	1062
5.87.2.6	<code>_M_revalidate_singular</code>	1062
5.87.2.7	<code>_M_swap</code>	1062
5.87.2.8	<code>_M_transfer_iter</code>	1063
5.87.3	Member Data Documentation	1063
5.87.3.1	<code>_M_const_iterators</code>	1063
5.87.3.2	<code>_M_iterators</code>	1063
5.87.3.3	<code>_M_version</code>	1063
5.88	<code>__gnu_debug::_Safe_sequence_base</code> Class Reference	1064
5.88.1	Detailed Description	1065
5.88.2	Constructor & Destructor Documentation	1065
5.88.2.1	<code>~_Safe_sequence_base</code>	1065
5.88.3	Member Function Documentation	1065
5.88.3.1	<code>_M_detach_all</code>	1065
5.88.3.2	<code>_M_detach_singular</code>	1065

5.88.3.3	<code>_M_get_mutex</code>	1065
5.88.3.4	<code>_M_invalidate_all</code>	1066
5.88.3.5	<code>_M_revalidate_singular</code>	1066
5.88.3.6	<code>_M_swap</code>	1066
5.88.4	Member Data Documentation	1066
5.88.4.1	<code>_M_const_iterators</code>	1066
5.88.4.2	<code>_M_iterators</code>	1067
5.88.4.3	<code>_M_version</code>	1067
5.89	<code>__gnu_debug::basic_string<_CharT, _Traits, _Allocator > Class</code>	
	Template Reference	1068
5.89.1	Detailed Description	1075
5.89.2	Constructor & Destructor Documentation	1075
5.89.2.1	<code>basic_string</code>	1075
5.89.2.2	<code>basic_string</code>	1075
5.89.2.3	<code>basic_string</code>	1076
5.89.2.4	<code>basic_string</code>	1076
5.89.2.5	<code>~basic_string</code>	1076
5.89.3	Member Function Documentation	1077
5.89.3.1	<code>_M_detach_all</code>	1077
5.89.3.2	<code>_M_detach_singular</code>	1077
5.89.3.3	<code>_M_get_mutex</code>	1077
5.89.3.4	<code>_M_invalidate_all</code>	1077
5.89.3.5	<code>_M_invalidate_if</code>	1078
5.89.3.6	<code>_M_revalidate_singular</code>	1078
5.89.3.7	<code>_M_swap</code>	1078
5.89.3.8	<code>_M_transfer_iter</code>	1078
5.89.3.9	<code>append</code>	1078
5.89.3.10	<code>append</code>	1079
5.89.3.11	<code>append</code>	1079
5.89.3.12	<code>append</code>	1079
5.89.3.13	<code>append</code>	1080

5.89.3.14	append	1080
5.89.3.15	append	1081
5.89.3.16	assign	1081
5.89.3.17	assign	1081
5.89.3.18	assign	1082
5.89.3.19	assign	1082
5.89.3.20	assign	1083
5.89.3.21	assign	1083
5.89.3.22	assign	1084
5.89.3.23	assign	1084
5.89.3.24	at	1085
5.89.3.25	at	1085
5.89.3.26	begin	1086
5.89.3.27	begin	1086
5.89.3.28	c_str	1086
5.89.3.29	capacity	1086
5.89.3.30	cbegin	1087
5.89.3.31	cend	1087
5.89.3.32	clear	1087
5.89.3.33	compare	1087
5.89.3.34	compare	1088
5.89.3.35	compare	1088
5.89.3.36	compare	1089
5.89.3.37	compare	1089
5.89.3.38	compare	1090
5.89.3.39	copy	1091
5.89.3.40	crbegin	1091
5.89.3.41	crend	1091
5.89.3.42	data	1091
5.89.3.43	empty	1092
5.89.3.44	end	1092

5.89.3.45 end	1092
5.89.3.46 erase	1092
5.89.3.47 erase	1093
5.89.3.48 erase	1093
5.89.3.49 find	1094
5.89.3.50 find	1094
5.89.3.51 find	1095
5.89.3.52 find	1095
5.89.3.53 find_first_not_of	1095
5.89.3.54 find_first_not_of	1096
5.89.3.55 find_first_not_of	1096
5.89.3.56 find_first_not_of	1097
5.89.3.57 find_first_of	1097
5.89.3.58 find_first_of	1098
5.89.3.59 find_first_of	1098
5.89.3.60 find_first_of	1098
5.89.3.61 find_last_not_of	1099
5.89.3.62 find_last_not_of	1099
5.89.3.63 find_last_not_of	1100
5.89.3.64 find_last_not_of	1100
5.89.3.65 find_last_of	1101
5.89.3.66 find_last_of	1101
5.89.3.67 find_last_of	1101
5.89.3.68 find_last_of	1102
5.89.3.69 get_allocator	1102
5.89.3.70 insert	1102
5.89.3.71 insert	1103
5.89.3.72 insert	1104
5.89.3.73 insert	1104
5.89.3.74 insert	1105
5.89.3.75 insert	1105

5.89.3.76 insert	1106
5.89.3.77 insert	1106
5.89.3.78 insert	1107
5.89.3.79 length	1107
5.89.3.80 max_size	1107
5.89.3.81 operator+=	1108
5.89.3.82 operator+=	1108
5.89.3.83 operator+=	1108
5.89.3.84 operator+=	1109
5.89.3.85 operator=	1109
5.89.3.86 operator=	1110
5.89.3.87 operator=	1110
5.89.3.88 operator[]	1110
5.89.3.89 operator[]	1111
5.89.3.90 push_back	1111
5.89.3.91 rbegin	1112
5.89.3.92 rbegin	1112
5.89.3.93 rend	1112
5.89.3.94 rend	1113
5.89.3.95 replace	1113
5.89.3.96 replace	1113
5.89.3.97 replace	1114
5.89.3.98 replace	1115
5.89.3.99 replace	1115
5.89.3.100replace	1116
5.89.3.101replace	1116
5.89.3.102replace	1117
5.89.3.103replace	1118
5.89.3.104replace	1118
5.89.3.105replace	1119
5.89.3.106reserve	1119

5.89.3.107	<code>resize</code>	1120
5.89.3.108	<code>resize</code>	1120
5.89.3.109	<code>find</code>	1121
5.89.3.110	<code>find</code>	1121
5.89.3.111	<code>lrfind</code>	1121
5.89.3.112	<code>rfind</code>	1122
5.89.3.113	<code>shrink_to_fit</code>	1122
5.89.3.114	<code>size</code>	1122
5.89.3.115	<code>substr</code>	1123
5.89.3.116	<code>swap</code>	1123
5.89.4	Member Data Documentation	1123
5.89.4.1	<code>_M_const_iterators</code>	1123
5.89.4.2	<code>_M_iterators</code>	1124
5.89.4.3	<code>_M_version</code>	1124
5.89.4.4	<code>npos</code>	1124
5.90	<code>__gnu_parallel::__accumulate_binop_reduct<_BinOp></code> Struct Template Reference	1125
5.90.1	Detailed Description	1125
5.91	<code>__gnu_parallel::__accumulate_selector<_It></code> Struct Template Reference	1126
5.91.1	Detailed Description	1126
5.91.2	Member Function Documentation	1126
5.91.2.1	<code>operator()</code>	1126
5.91.3	Member Data Documentation	1127
5.91.3.1	<code>_M_finish_iterator</code>	1127
5.92	<code>__gnu_parallel::__adjacent_difference_selector<_It></code> Struct Template Reference	1128
5.92.1	Detailed Description	1128
5.92.2	Member Data Documentation	1128
5.92.2.1	<code>_M_finish_iterator</code>	1128
5.93	<code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	1130
5.93.1	Detailed Description	1130

5.93.2	Member Function Documentation	1130
5.93.2.1	_M_sequential_algorithm	1130
5.93.2.2	operator()	1131
5.94	__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . .	1132
5.94.1	Detailed Description	1133
5.94.2	Member Typedef Documentation	1133
5.94.2.1	argument_type	1133
5.94.2.2	result_type	1133
5.95	__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . .	1134
5.95.1	Detailed Description	1135
5.95.2	Member Typedef Documentation	1135
5.95.2.1	argument_type	1135
5.95.2.2	result_type	1135
5.96	__gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference	1136
5.96.1	Detailed Description	1136
5.96.2	Member Function Documentation	1136
5.96.2.1	operator()	1136
5.96.3	Member Data Documentation	1137
5.96.3.1	_M_finish_iterator	1137
5.97	__gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference	1138
5.97.1	Detailed Description	1138
5.97.2	Member Function Documentation	1138
5.97.2.1	operator()	1138
5.97.3	Member Data Documentation	1139
5.97.3.1	_M_finish_iterator	1139
5.98	__gnu_parallel::__fill_selector< _It > Struct Template Reference . . .	1140
5.98.1	Detailed Description	1140
5.98.2	Member Function Documentation	1140

5.98.2.1	operator()	1140
5.98.3	Member Data Documentation	1141
5.98.3.1	_M_finish_iterator	1141
5.99	__gnu_parallel::__find_first_of_selector<_FIterator> Struct Template Reference	1142
5.99.1	Detailed Description	1142
5.99.2	Member Function Documentation	1143
5.99.2.1	_M_sequential_algorithm	1143
5.99.2.2	operator()	1143
5.100	__gnu_parallel::__find_if_selector Struct Reference	1144
5.100.1	Detailed Description	1144
5.100.2	Member Function Documentation	1144
5.100.2.1	_M_sequential_algorithm	1144
5.100.2.2	operator()	1145
5.101	__gnu_parallel::__for_each_selector<_It> Struct Template Reference	1146
5.101.1	Detailed Description	1146
5.101.2	Member Function Documentation	1146
5.101.2.1	operator()	1146
5.101.3	Member Data Documentation	1147
5.101.3.1	_M_finish_iterator	1147
5.102	__gnu_parallel::__generate_selector<_It> Struct Template Reference	1148
5.102.1	Detailed Description	1148
5.102.2	Member Function Documentation	1148
5.102.2.1	operator()	1148
5.102.3	Member Data Documentation	1149
5.102.3.1	_M_finish_iterator	1149
5.103	__gnu_parallel::__generic_find_selector Struct Reference	1150
5.103.1	Detailed Description	1150
5.104	__gnu_parallel::__generic_for_each_selector<_It> Struct Template Reference	1151
5.104.1	Detailed Description	1152
5.104.2	Member Data Documentation	1152

5.104.2.1 <code>_M_finish_iterator</code>	1152
5.105 <code>__gnu_parallel::__identity_selector<_It></code> Struct Template Reference	1153
5.105.1 Detailed Description	1153
5.105.2 Member Function Documentation	1153
5.105.2.1 <code>operator()</code>	1153
5.105.3 Member Data Documentation	1154
5.105.3.1 <code>_M_finish_iterator</code>	1154
5.106 <code>__gnu_parallel::__inner_product_selector<_It, _It2, _Tp></code> Struct Template Reference	1155
5.106.1 Detailed Description	1155
5.106.2 Constructor & Destructor Documentation	1156
5.106.2.1 <code>__inner_product_selector</code>	1156
5.106.3 Member Function Documentation	1156
5.106.3.1 <code>operator()</code>	1156
5.106.4 Member Data Documentation	1156
5.106.4.1 <code>__begin1_iterator</code>	1156
5.106.4.2 <code>__begin2_iterator</code>	1157
5.106.4.3 <code>_M_finish_iterator</code>	1157
5.107 <code>__gnu_parallel::__max_element_reduct<_Compare, _It></code> Struct Template Reference	1158
5.107.1 Detailed Description	1158
5.108 <code>__gnu_parallel::__min_element_reduct<_Compare, _It></code> Struct Template Reference	1159
5.108.1 Detailed Description	1159
5.109 <code>__gnu_parallel::__mismatch_selector</code> Struct Reference	1160
5.109.1 Detailed Description	1160
5.109.2 Member Function Documentation	1160
5.109.2.1 <code>_M_sequential_algorithm</code>	1160
5.109.2.2 <code>operator()</code>	1161
5.110 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch<__-</code> <code>sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	1162
5.110.1 Detailed Description	1162

5.111 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference</code>	1163
5.111.1 Detailed Description	1163
5.112 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference</code>	1164
5.112.1 Detailed Description	1164
5.113 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference</code>	1165
5.113.1 Detailed Description	1165
5.114 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference</code>	1166
5.114.1 Detailed Description	1166
5.115 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference</code>	1167
5.115.1 Detailed Description	1167
5.116 <code>__gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference</code>	1168
5.116.1 Detailed Description	1168
5.116.2 Constructor & Destructor Documentation	1169
5.116.2.1 <code>__replace_if_selector</code>	1169
5.116.3 Member Function Documentation	1169
5.116.3.1 <code>operator()</code>	1169
5.116.4 Member Data Documentation	1169
5.116.4.1 <code>__new_val</code>	1169
5.116.4.2 <code>_M_finish_iterator</code>	1170
5.117 <code>__gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference</code>	1171
5.117.1 Detailed Description	1171
5.117.2 Constructor & Destructor Documentation	1171
5.117.2.1 <code>__replace_selector</code>	1171

5.117.3 Member Function Documentation	1172
5.117.3.1 operator()	1172
5.117.4 Member Data Documentation	1172
5.117.4.1 __new_val	1172
5.117.4.2 _M_finish_iterator	1172
5.118 __gnu_parallel::__transform1_selector< _It > Struct Template Reference	1173
5.118.1 Detailed Description	1173
5.118.2 Member Function Documentation	1173
5.118.2.1 operator()	1173
5.118.3 Member Data Documentation	1174
5.118.3.1 _M_finish_iterator	1174
5.119 __gnu_parallel::__transform2_selector< _It > Struct Template Reference	1175
5.119.1 Detailed Description	1175
5.119.2 Member Function Documentation	1175
5.119.2.1 operator()	1175
5.119.3 Member Data Documentation	1176
5.119.3.1 _M_finish_iterator	1176
5.120 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference	1177
5.120.1 Detailed Description	1178
5.120.2 Member Typedef Documentation	1178
5.120.2.1 argument_type	1178
5.120.2.2 result_type	1178
5.121 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference	1179
5.121.1 Detailed Description	1179
5.121.2 Constructor & Destructor Documentation	1180
5.121.2.1 _DRandomShufflingGlobalData	1180
5.121.3 Member Data Documentation	1180
5.121.3.1 _M_bin_proc	1180

5.121.3.2	<code>_M_dist</code>	1180
5.121.3.3	<code>_M_num_bins</code>	1180
5.121.3.4	<code>_M_num_bits</code>	1181
5.121.3.5	<code>_M_source</code>	1181
5.121.3.6	<code>_M_starts</code>	1181
5.121.3.7	<code>_M_temporaries</code>	1181
5.122	<code>__gnu_parallel::_DRSSorterPU<_RAIter, RandomNumberGenerator></code> Struct Template Reference	1182
5.122.1	Detailed Description	1182
5.122.2	Member Data Documentation	1182
5.122.2.1	<code>__bins_end</code>	1182
5.122.2.2	<code>_M_bins_begin</code>	1182
5.122.2.3	<code>_M_num_threads</code>	1183
5.122.2.4	<code>_M_sd</code>	1183
5.122.2.5	<code>_M_seed</code>	1183
5.123	<code>__gnu_parallel::_DummyReduct</code> Struct Reference	1184
5.123.1	Detailed Description	1184
5.124	<code>__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare></code> Class Template Reference	1185
5.124.1	Detailed Description	1185
5.124.2	Member Typedef Documentation	1186
5.124.2.1	<code>first_argument_type</code>	1186
5.124.2.2	<code>result_type</code>	1186
5.124.2.3	<code>second_argument_type</code>	1186
5.125	<code>__gnu_parallel::_EqualTo<_T1, _T2></code> Struct Template Reference	1187
5.125.1	Detailed Description	1187
5.125.2	Member Typedef Documentation	1188
5.125.2.1	<code>first_argument_type</code>	1188
5.125.2.2	<code>result_type</code>	1188
5.125.2.3	<code>second_argument_type</code>	1188
5.126	<code>__gnu_parallel::_GuardedIterator<_RAIter, _Compare></code> Class Template Reference	1189

5.126.1 Detailed Description	1189
5.126.2 Constructor & Destructor Documentation	1189
5.126.2.1 <code>_GuardedIterator</code>	1189
5.126.3 Member Function Documentation	1190
5.126.3.1 <code>operator_RAIter</code>	1190
5.126.3.2 <code>operator*</code>	1190
5.126.3.3 <code>operator++</code>	1190
5.126.4 Friends And Related Function Documentation	1191
5.126.4.1 <code>operator<</code>	1191
5.126.4.2 <code>operator<=</code>	1191
5.127 <code>__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory ></code> Class Template Reference	1192
5.127.1 Detailed Description	1193
5.127.2 Member Typedef Documentation	1193
5.127.2.1 <code>first_type</code>	1193
5.127.2.2 <code>second_type</code>	1193
5.127.3 Member Data Documentation	1193
5.127.3.1 <code>first</code>	1193
5.127.3.2 <code>second</code>	1194
5.128 <code>__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory ></code> Class Template Reference	1195
5.128.1 Detailed Description	1195
5.129 <code>__gnu_parallel::_Job< _DifferenceTp ></code> Struct Template Reference	1197
5.129.1 Detailed Description	1197
5.129.2 Member Data Documentation	1197
5.129.2.1 <code>_M_first</code>	1197
5.129.2.2 <code>_M_last</code>	1197
5.129.2.3 <code>_M_load</code>	1198
5.130 <code>__gnu_parallel::_Less< _T1, _T2 ></code> Struct Template Reference	1199
5.130.1 Detailed Description	1199
5.130.2 Member Typedef Documentation	1200
5.130.2.1 <code>first_argument_type</code>	1200

5.130.2.2	result_type	1200
5.130.2.3	second_argument_type	1200
5.131	<code>__gnu_parallel::_Lexicographic< _T1, _T2, _Compare ></code> Class Template Reference	1201
5.131.1	Detailed Description	1201
5.131.2	Member Typedef Documentation	1202
5.131.2.1	first_argument_type	1202
5.131.2.2	result_type	1202
5.131.2.3	second_argument_type	1202
5.132	<code>__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare ></code> Class Template Reference	1203
5.132.1	Detailed Description	1203
5.132.2	Member Typedef Documentation	1204
5.132.2.1	first_argument_type	1204
5.132.2.2	result_type	1204
5.132.2.3	second_argument_type	1204
5.133	<code>__gnu_parallel::_LoserTree< __stable, _Tp, _Compare ></code> Class Template Reference	1205
5.133.1	Detailed Description	1205
5.133.2	Member Function Documentation	1206
5.133.2.1	<code>__delete_min_insert</code>	1206
5.133.2.2	<code>__get_min_source</code>	1206
5.133.2.3	<code>__insert_start</code>	1206
5.133.3	Member Data Documentation	1207
5.133.3.1	<code>_M_comp</code>	1207
5.133.3.2	<code>_M_first_insert</code>	1207
5.133.3.3	<code>_M_log_k</code>	1207
5.133.3.4	<code>_M_losers</code>	1207
5.134	<code>__gnu_parallel::_LoserTree< false, _Tp, _Compare ></code> Class Template Reference	1209
5.134.1	Detailed Description	1209
5.134.2	Member Function Documentation	1210

5.134.2.1	<code>__delete_min_insert</code>	1210
5.134.2.2	<code>__get_min_source</code>	1210
5.134.2.3	<code>__init_winner</code>	1210
5.134.2.4	<code>__insert_start</code>	1211
5.134.3	Member Data Documentation	1211
5.134.3.1	<code>_M_comp</code>	1211
5.134.3.2	<code>_M_first_insert</code>	1211
5.134.3.3	<code>_M_log_k</code>	1212
5.134.3.4	<code>_M_losers</code>	1212
5.135	<code>__gnu_parallel::_LoserTreeBase< _Tp, _Compare ></code> Class Template Reference	1213
5.135.1	Detailed Description	1213
5.135.2	Constructor & Destructor Documentation	1214
5.135.2.1	<code>_LoserTreeBase</code>	1214
5.135.2.2	<code>~_LoserTreeBase</code>	1214
5.135.3	Member Function Documentation	1215
5.135.3.1	<code>__get_min_source</code>	1215
5.135.3.2	<code>__insert_start</code>	1215
5.135.4	Member Data Documentation	1215
5.135.4.1	<code>_M_comp</code>	1215
5.135.4.2	<code>_M_first_insert</code>	1216
5.135.4.3	<code>_M_log_k</code>	1216
5.135.4.4	<code>_M_losers</code>	1216
5.136	<code>__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser</code> Struct Reference	1217
5.136.1	Detailed Description	1217
5.136.2	Member Data Documentation	1217
5.136.2.1	<code>_M_key</code>	1217
5.136.2.2	<code>_M_source</code>	1217
5.136.2.3	<code>_M_sup</code>	1218
5.137	<code>__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare ></code> Class Template Reference	1219

5.137.1 Detailed Description	1219
5.138__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference	1221
5.138.1 Detailed Description	1221
5.139__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference	1223
5.139.1 Detailed Description	1223
5.140__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference	1225
5.140.1 Detailed Description	1225
5.141__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _- Compare > Class Template Reference	1226
5.141.1 Detailed Description	1226
5.142__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference	1228
5.142.1 Detailed Description	1228
5.143__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference	1230
5.143.1 Detailed Description	1230
5.144__gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference	1231
5.144.1 Detailed Description	1231
5.144.2 Member Data Documentation	1231
5.144.2.1 _M_use_pointer	1231
5.145__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference	1233
5.145.1 Detailed Description	1233
5.146__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference	1235
5.146.1 Detailed Description	1235
5.147__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference	1237
5.147.1 Detailed Description	1237
5.148__gnu_parallel::_Multiplies< _Tp1, _Tp2 > Struct Template Reference	1239
5.148.1 Detailed Description	1239

5.148.2 Member Typedef Documentation	1240
5.148.2.1 first_argument_type	1240
5.148.2.2 result_type	1240
5.148.2.3 second_argument_type	1240
5.149 __gnu_parallel::_Nothing Struct Reference	1241
5.149.1 Detailed Description	1241
5.149.2 Member Function Documentation	1241
5.149.2.1 operator()	1241
5.150 __gnu_parallel::_Piece<_DifferenceTp> Struct Template Reference	1242
5.150.1 Detailed Description	1242
5.150.2 Member Data Documentation	1242
5.150.2.1 _M_begin	1242
5.150.2.2 _M_end	1242
5.151 __gnu_parallel::_Plus<_Tp1, _Tp2> Struct Template Reference	1243
5.151.1 Detailed Description	1243
5.151.2 Member Typedef Documentation	1244
5.151.2.1 first_argument_type	1244
5.151.2.2 result_type	1244
5.151.2.3 second_argument_type	1244
5.152 __gnu_parallel::_PMWMSortingData<_RAIter> Struct Template Reference	1245
5.152.1 Detailed Description	1245
5.152.2 Member Data Documentation	1245
5.152.2.1 _M_num_threads	1245
5.152.2.2 _M_offsets	1246
5.152.2.3 _M_pieces	1246
5.152.2.4 _M_samples	1246
5.152.2.5 _M_source	1246
5.152.2.6 _M_starts	1246
5.152.2.7 _M_temporary	1247
5.153 __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp> Class Template Reference	1248

5.153.1 Detailed Description	1248
5.153.2 Constructor & Destructor Documentation	1248
5.153.2.1 _PseudoSequence	1248
5.153.3 Member Function Documentation	1249
5.153.3.1 begin	1249
5.153.3.2 end	1249
5.154 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference	1250
5.154.1 Detailed Description	1250
5.155 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Ref- erence	1251
5.155.1 Detailed Description	1251
5.155.2 Member Typedef Documentation	1251
5.155.2.1 _Piece	1251
5.155.3 Constructor & Destructor Documentation	1252
5.155.3.1 _QSBThreadLocal	1252
5.155.4 Member Data Documentation	1252
5.155.4.1 _M_elements_leftover	1252
5.155.4.2 _M_global	1252
5.155.4.3 _M_initial	1252
5.155.4.4 _M_leftover_parts	1253
5.155.4.5 _M_num_threads	1253
5.156 __gnu_parallel::_RandomNumber Class Reference	1254
5.156.1 Detailed Description	1254
5.156.2 Constructor & Destructor Documentation	1254
5.156.2.1 _RandomNumber	1254
5.156.2.2 _RandomNumber	1254
5.156.3 Member Function Documentation	1255
5.156.3.1 __genrand_bits	1255
5.156.3.2 operator()	1255
5.156.3.3 operator()	1255

5.157	__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class	
	Template Reference	1256
5.157.1	Detailed Description	1256
5.157.2	Constructor & Destructor Documentation	1257
5.157.2.1	_RestrictedBoundedConcurrentQueue	1257
5.157.2.2	~_RestrictedBoundedConcurrentQueue	1257
5.157.3	Member Function Documentation	1257
5.157.3.1	pop_back	1257
5.157.3.2	pop_front	1257
5.157.3.3	push_front	1258
5.158	__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct	
	Template Reference	1259
5.158.1	Detailed Description	1259
5.159	__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct	
	Template Reference	1260
5.159.1	Detailed Description	1260
5.160	__gnu_parallel::_Settings Struct Reference	1261
5.160.1	Detailed Description	1262
5.160.2	Member Function Documentation	1262
5.160.2.1	__attribute__	1262
5.160.2.2	set	1262
5.160.3	Member Data Documentation	1263
5.160.3.1	accumulate_minimal_n	1263
5.160.3.2	adjacent_difference_minimal_n	1263
5.160.3.3	cache_line_size	1263
5.160.3.4	count_minimal_n	1263
5.160.3.5	fill_minimal_n	1263
5.160.3.6	find_increasing_factor	1263
5.160.3.7	find_initial_block_size	1264
5.160.3.8	find_maximum_block_size	1264
5.160.3.9	find_sequential_search_size	1264
5.160.3.10	for_each_minimal_n	1264

5.160.3.1	generate_minimal_n	1264
5.160.3.12	L1_cache_size	1264
5.160.3.13	L2_cache_size	1265
5.160.3.14	max_element_minimal_n	1265
5.160.3.15	merge_minimal_n	1265
5.160.3.16	merge_oversampling	1265
5.160.3.17	min_element_minimal_n	1265
5.160.3.18	multiway_merge_minimal_k	1265
5.160.3.19	multiway_merge_minimal_n	1265
5.160.3.20	multiway_merge_oversampling	1266
5.160.3.2	nth_element_minimal_n	1266
5.160.3.22	partial_sort_minimal_n	1266
5.160.3.23	partial_sum_dilation	1266
5.160.3.24	partial_sum_minimal_n	1266
5.160.3.25	partition_chunk_share	1266
5.160.3.26	partition_chunk_size	1267
5.160.3.27	partition_minimal_n	1267
5.160.3.28	qsb_steals	1267
5.160.3.29	random_shuffle_minimal_n	1267
5.160.3.30	replace_minimal_n	1267
5.160.3.3	search_minimal_n	1267
5.160.3.32	set_difference_minimal_n	1267
5.160.3.33	set_intersection_minimal_n	1268
5.160.3.34	set_symmetric_difference_minimal_n	1268
5.160.3.35	set_union_minimal_n	1268
5.160.3.36	sort_minimal_n	1268
5.160.3.37	sort_mwms_oversampling	1268
5.160.3.38	sort_qs_num_samples_preset	1268
5.160.3.39	sort_qsb_base_case_maximal_n	1268
5.160.3.40	TLB_size	1269
5.160.3.4	transform_minimal_n	1269

5.160.3.4	<code>unique_copy_minimal_n</code>	1269
5.161	<code>__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _- SortingPlacesIterator ></code> Struct Template Reference	1270
5.161.1	Detailed Description	1270
5.162	<code>__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _- SortingPlacesIterator ></code> Struct Template Reference	1271
5.162.1	Detailed Description	1271
5.163	<code>__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _- SortingPlacesIterator ></code> Struct Template Reference	1272
5.163.1	Detailed Description	1272
5.164	<code>__gnu_parallel::_VoidFunctor< _ValueTp ></code> Class Template Reference	1273
5.164.1	Detailed Description	1273
5.165	<code>__gnu_parallel::balanced_quicksort_tag</code> Struct Reference	1274
5.165.1	Detailed Description	1274
5.165.2	Member Function Documentation	1274
5.165.2.1	<code>__get_num_threads</code>	1274
5.165.2.2	<code>set_num_threads</code>	1275
5.166	<code>__gnu_parallel::balanced_tag</code> Struct Reference	1276
5.166.1	Detailed Description	1276
5.166.2	Member Function Documentation	1276
5.166.2.1	<code>__get_num_threads</code>	1276
5.166.2.2	<code>set_num_threads</code>	1277
5.167	<code>__gnu_parallel::constant_size_blocks_tag</code> Struct Reference	1278
5.167.1	Detailed Description	1278
5.168	<code>__gnu_parallel::default_parallel_tag</code> Struct Reference	1279
5.168.1	Detailed Description	1279
5.168.2	Member Function Documentation	1279
5.168.2.1	<code>__get_num_threads</code>	1279
5.168.2.2	<code>set_num_threads</code>	1280
5.169	<code>__gnu_parallel::equal_split_tag</code> Struct Reference	1281
5.169.1	Detailed Description	1281
5.170	<code>__gnu_parallel::exact_tag</code> Struct Reference	1282

5.170.1 Detailed Description	1282
5.170.2 Member Function Documentation	1282
5.170.2.1 __get_num_threads	1282
5.170.2.2 set_num_threads	1283
5.171 __gnu_parallel::find_tag Struct Reference	1284
5.171.1 Detailed Description	1284
5.172 __gnu_parallel::growing_blocks_tag Struct Reference	1285
5.172.1 Detailed Description	1285
5.173 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	1286
5.173.1 Detailed Description	1286
5.173.2 Member Function Documentation	1286
5.173.2.1 __get_num_threads	1286
5.173.2.2 set_num_threads	1287
5.174 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	1288
5.174.1 Detailed Description	1288
5.174.2 Member Function Documentation	1288
5.174.2.1 __get_num_threads	1288
5.174.2.2 set_num_threads	1289
5.175 __gnu_parallel::multiway_mergesort_tag Struct Reference	1290
5.175.1 Detailed Description	1290
5.175.2 Member Function Documentation	1290
5.175.2.1 __get_num_threads	1290
5.175.2.2 set_num_threads	1291
5.176 __gnu_parallel::omp_loop_static_tag Struct Reference	1292
5.176.1 Detailed Description	1292
5.176.2 Member Function Documentation	1292
5.176.2.1 __get_num_threads	1292
5.176.2.2 set_num_threads	1293
5.177 __gnu_parallel::omp_loop_tag Struct Reference	1294
5.177.1 Detailed Description	1294
5.177.2 Member Function Documentation	1294

5.177.2.1	<code>__get_num_threads</code>	1294
5.177.2.2	<code>set_num_threads</code>	1295
5.178	<code>gnu_parallel::parallel_tag</code> Struct Reference	1296
5.178.1	Detailed Description	1297
5.178.2	Constructor & Destructor Documentation	1297
5.178.2.1	<code>parallel_tag</code>	1297
5.178.2.2	<code>parallel_tag</code>	1297
5.178.3	Member Function Documentation	1297
5.178.3.1	<code>__get_num_threads</code>	1297
5.178.3.2	<code>set_num_threads</code>	1298
5.179	<code>gnu_parallel::quicksort_tag</code> Struct Reference	1299
5.179.1	Detailed Description	1299
5.179.2	Member Function Documentation	1299
5.179.2.1	<code>__get_num_threads</code>	1299
5.179.2.2	<code>set_num_threads</code>	1300
5.180	<code>gnu_parallel::sampling_tag</code> Struct Reference	1301
5.180.1	Detailed Description	1301
5.180.2	Member Function Documentation	1301
5.180.2.1	<code>__get_num_threads</code>	1301
5.180.2.2	<code>set_num_threads</code>	1302
5.181	<code>gnu_parallel::sequential_tag</code> Struct Reference	1303
5.181.1	Detailed Description	1303
5.182	<code>gnu_parallel::unbalanced_tag</code> Struct Reference	1304
5.182.1	Detailed Description	1304
5.182.2	Member Function Documentation	1304
5.182.2.1	<code>__get_num_threads</code>	1304
5.182.2.2	<code>set_num_threads</code>	1305
5.183	<code>gnu_pbds::associative_container_tag</code> Struct Reference	1306
5.183.1	Detailed Description	1306
5.184	<code>gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator ></code> Class Template Reference	1307

5.184.1 Detailed Description	1308
5.185__gnu_pbds::basic_hash_tag Struct Reference	1309
5.185.1 Detailed Description	1309
5.186__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator > Class Template Reference	1310
5.186.1 Detailed Description	1311
5.187__gnu_pbds::basic_tree_tag Struct Reference	1312
5.187.1 Detailed Description	1312
5.188__gnu_pbds::binary_heap_tag Struct Reference	1313
5.188.1 Detailed Description	1313
5.189__gnu_pbds::binomial_heap_tag Struct Reference	1314
5.189.1 Detailed Description	1314
5.190__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1315
5.190.1 Detailed Description	1316
5.191__gnu_pbds::cc_hash_tag Struct Reference	1318
5.191.1 Detailed Description	1318
5.192__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator > Class Template Reference	1319
5.192.1 Detailed Description	1320
5.193__gnu_pbds::container_tag Struct Reference	1321
5.193.1 Detailed Description	1321
5.194__gnu_pbds::container_traits< Cntnr > Struct Template Reference	1322
5.194.1 Detailed Description	1322
5.195__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false > Struct Template Reference	1323
5.195.1 Detailed Description	1323
5.196__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true > Struct Template Reference	1324
5.196.1 Detailed Description	1324
5.197__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false > Struct Template Reference	1325

5.197.1 Detailed Description	1325
5.198__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true > Struct Template Reference	1326
5.198.1 Detailed Description	1326
5.199__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1327
5.199.1 Detailed Description	1329
5.200__gnu_pbds::gp_hash_tag Struct Reference	1330
5.200.1 Detailed Description	1330
5.201__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator > Class Template Reference	1331
5.201.1 Detailed Description	1332
5.202__gnu_pbds::list_update_tag Struct Reference	1333
5.202.1 Detailed Description	1333
5.203__gnu_pbds::null_mapped_type Struct Reference	1334
5.203.1 Detailed Description	1334
5.204__gnu_pbds::ov_tree_tag Struct Reference	1335
5.204.1 Detailed Description	1335
5.205__gnu_pbds::pairing_heap_tag Struct Reference	1336
5.205.1 Detailed Description	1336
5.206__gnu_pbds::pat_trie_tag Struct Reference	1337
5.206.1 Detailed Description	1337
5.207__gnu_pbds::priority_queue_tag Struct Reference	1338
5.207.1 Detailed Description	1338
5.208__gnu_pbds::rb_tree_tag Struct Reference	1339
5.208.1 Detailed Description	1339
5.209__gnu_pbds::rc_binomial_heap_tag Struct Reference	1340
5.209.1 Detailed Description	1340
5.210__gnu_pbds::sequence_tag Struct Reference	1341
5.210.1 Detailed Description	1341
5.211__gnu_pbds::splay_tree_tag Struct Reference	1342

5.211.1 Detailed Description	1342
5.212__gnu_pbds::string_tag Struct Reference	1343
5.212.1 Detailed Description	1343
5.213__gnu_pbds::thin_heap_tag Struct Reference	1344
5.213.1 Detailed Description	1344
5.214__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator > Class Template Reference	1345
5.214.1 Detailed Description	1346
5.215__gnu_pbds::tree_tag Struct Reference	1347
5.215.1 Detailed Description	1347
5.216__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator > Class Template Reference	1348
5.216.1 Detailed Description	1349
5.217__gnu_pbds::trie_tag Struct Reference	1350
5.217.1 Detailed Description	1350
5.218__gnu_profile::__container_size_info Class Reference	1351
5.218.1 Detailed Description	1352
5.219__gnu_profile::__container_size_stack_info Class Reference	1353
5.219.1 Detailed Description	1354
5.220__gnu_profile::__hashfunc_info Class Reference	1355
5.220.1 Detailed Description	1355
5.221__gnu_profile::__hashfunc_stack_info Class Reference	1357
5.221.1 Detailed Description	1357
5.222__gnu_profile::__list2vector_info Class Reference	1359
5.222.1 Detailed Description	1360
5.223__gnu_profile::__map2umap_info Class Reference	1361
5.223.1 Detailed Description	1362
5.224__gnu_profile::__map2umap_stack_info Class Reference	1363
5.224.1 Detailed Description	1364
5.225__gnu_profile::__object_info_base Class Reference	1365
5.225.1 Detailed Description	1365
5.226__gnu_profile::__reentrance_guard Struct Reference	1366

5.226.1 Detailed Description	1366
5.227 <code>__gnu_profile::__stack_hash</code> Class Reference	1367
5.227.1 Detailed Description	1367
5.228 <code>__gnu_profile::__stack_info_base< __object_info ></code> Class Template Reference	1368
5.228.1 Detailed Description	1368
5.229 <code>__gnu_profile::__trace_base< __object_info, __stack_info ></code> Class Template Reference	1369
5.229.1 Detailed Description	1369
5.230 <code>__gnu_profile::__trace_container_size</code> Class Reference	1370
5.230.1 Detailed Description	1370
5.231 <code>__gnu_profile::__trace_hash_func</code> Class Reference	1371
5.231.1 Detailed Description	1372
5.232 <code>__gnu_profile::__trace_hashtable_size</code> Class Reference	1373
5.232.1 Detailed Description	1373
5.233 <code>__gnu_profile::__trace_map2umap</code> Class Reference	1374
5.233.1 Detailed Description	1374
5.234 <code>__gnu_profile::__trace_vector_size</code> Class Reference	1376
5.234.1 Detailed Description	1376
5.235 <code>__gnu_profile::__trace_vector_to_list</code> Class Reference	1377
5.235.1 Detailed Description	1378
5.236 <code>__gnu_profile::__vector2list_info</code> Class Reference	1379
5.236.1 Detailed Description	1380
5.237 <code>__gnu_profile::__vector2list_stack_info</code> Class Reference	1381
5.237.1 Detailed Description	1382
5.238 <code>__gnu_profile::__warning_data</code> Struct Reference	1383
5.238.1 Detailed Description	1383
5.239 <code>std::__basic_future< _Res ></code> Class Template Reference	1384
5.239.1 Detailed Description	1385
5.239.2 Member Function Documentation	1385
5.239.2.1 <code>_M_get_result</code>	1385

5.240	<code>std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class</code>	
	Template Reference	1386
5.240.1	Detailed Description	1387
5.240.2	Member Function Documentation	1388
	5.240.2.1 <code>do_out</code>	1388
	5.240.2.2 <code>in</code>	1388
	5.240.2.3 <code>out</code>	1389
	5.240.2.4 <code>unshift</code>	1390
5.241	<code>std::__ctype_abstract_base< _CharT > Class Template Reference</code> . .	1392
	5.241.1 Detailed Description	1394
	5.241.2 Member Typedef Documentation	1394
	5.241.2.1 <code>char_type</code>	1394
	5.241.3 Member Function Documentation	1395
	5.241.3.1 <code>do_is</code>	1395
	5.241.3.2 <code>do_is</code>	1395
	5.241.3.3 <code>do_narrow</code>	1396
	5.241.3.4 <code>do_narrow</code>	1396
	5.241.3.5 <code>do_scan_is</code>	1397
	5.241.3.6 <code>do_scan_not</code>	1397
	5.241.3.7 <code>do_tolower</code>	1398
	5.241.3.8 <code>do_tolower</code>	1398
	5.241.3.9 <code>do_toupper</code>	1399
	5.241.3.10 <code>do_toupper</code>	1399
	5.241.3.11 <code>do_widen</code>	1400
	5.241.3.12 <code>do_widen</code>	1400
	5.241.3.13 <code>is</code>	1401
	5.241.3.14 <code>is</code>	1401
	5.241.3.15 <code>narrow</code>	1401
	5.241.3.16 <code>narrow</code>	1402
	5.241.3.17 <code>scan_is</code>	1402
	5.241.3.18 <code>scan_not</code>	1403

5.241.3.19	tolower	1403
5.241.3.20	tolower	1404
5.241.3.21	toupper	1404
5.241.3.22	toupper	1405
5.241.3.23	widen	1405
5.241.3.24	widen	1405
5.242	std::__debug::bitset< _Nb > Class Template Reference	1407
5.242.1	Detailed Description	1409
5.242.2	Member Function Documentation	1409
5.242.2.1	_M_detach_all	1409
5.242.2.2	_M_detach_singular	1409
5.242.2.3	_M_get_mutex	1409
5.242.2.4	_M_invalidate_all	1409
5.242.2.5	_M_revalidate_singular	1410
5.242.2.6	_M_swap	1410
5.242.3	Member Data Documentation	1410
5.242.3.1	_M_const_iterators	1410
5.242.3.2	_M_iterators	1410
5.242.3.3	_M_version	1411
5.243	std::__debug::deque< _Tp, _Allocator > Class Template Reference	1412
5.243.1	Detailed Description	1415
5.243.2	Member Function Documentation	1415
5.243.2.1	_M_detach_all	1415
5.243.2.2	_M_detach_singular	1415
5.243.2.3	_M_get_mutex	1415
5.243.2.4	_M_invalidate_all	1415
5.243.2.5	_M_invalidate_if	1416
5.243.2.6	_M_revalidate_singular	1416
5.243.2.7	_M_swap	1416
5.243.2.8	_M_transfer_iter	1416
5.243.3	Member Data Documentation	1416

5.243.3.1	_M_const_iterators	1416
5.243.3.2	_M_iterators	1417
5.243.3.3	_M_version	1417
5.244	std::__debug::list< _Tp, _Allocator > Class Template Reference	1418
5.244.1	Detailed Description	1421
5.244.2	Member Function Documentation	1421
5.244.2.1	_M_detach_all	1421
5.244.2.2	_M_detach_singular	1421
5.244.2.3	_M_get_mutex	1421
5.244.2.4	_M_invalidate_all	1422
5.244.2.5	_M_invalidate_if	1422
5.244.2.6	_M_revalidate_singular	1422
5.244.2.7	_M_swap	1422
5.244.2.8	_M_transfer_iter	1422
5.244.3	Member Data Documentation	1423
5.244.3.1	_M_const_iterators	1423
5.244.3.2	_M_iterators	1423
5.244.3.3	_M_version	1423
5.245	std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1424
5.245.1	Detailed Description	1426
5.245.2	Member Function Documentation	1426
5.245.2.1	_M_detach_all	1426
5.245.2.2	_M_detach_singular	1427
5.245.2.3	_M_get_mutex	1427
5.245.2.4	_M_invalidate_all	1427
5.245.2.5	_M_invalidate_if	1427
5.245.2.6	_M_revalidate_singular	1428
5.245.2.7	_M_swap	1428
5.245.2.8	_M_transfer_iter	1428
5.245.3	Member Data Documentation	1428

5.245.3.1	<code>_M_const_iterators</code>	1428
5.245.3.2	<code>_M_iterators</code>	1428
5.245.3.3	<code>_M_version</code>	1429
5.246	<code>std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class</code>	
	Template Reference	1430
5.246.1	Detailed Description	1432
5.246.2	Member Function Documentation	1432
5.246.2.1	<code>_M_detach_all</code>	1432
5.246.2.2	<code>_M_detach_singular</code>	1433
5.246.2.3	<code>_M_get_mutex</code>	1433
5.246.2.4	<code>_M_invalidate_all</code>	1433
5.246.2.5	<code>_M_invalidate_if</code>	1433
5.246.2.6	<code>_M_revalidate_singular</code>	1434
5.246.2.7	<code>_M_swap</code>	1434
5.246.2.8	<code>_M_transfer_iter</code>	1434
5.246.3	Member Data Documentation	1434
5.246.3.1	<code>_M_const_iterators</code>	1434
5.246.3.2	<code>_M_iterators</code>	1434
5.246.3.3	<code>_M_version</code>	1435
5.247	<code>std::__debug::multiset< _Key, _Compare, _Allocator > Class</code>	
	Template Reference	1436
5.247.1	Detailed Description	1438
5.247.2	Member Function Documentation	1438
5.247.2.1	<code>_M_detach_all</code>	1438
5.247.2.2	<code>_M_detach_singular</code>	1439
5.247.2.3	<code>_M_get_mutex</code>	1439
5.247.2.4	<code>_M_invalidate_all</code>	1439
5.247.2.5	<code>_M_invalidate_if</code>	1439
5.247.2.6	<code>_M_revalidate_singular</code>	1440
5.247.2.7	<code>_M_swap</code>	1440
5.247.2.8	<code>_M_transfer_iter</code>	1440
5.247.3	Member Data Documentation	1440

5.247.3.1	<code>_M_const_iterators</code>	1440
5.247.3.2	<code>_M_iterators</code>	1440
5.247.3.3	<code>_M_version</code>	1441
5.248	<code>std::__debug::set<_Key, _Compare, _Allocator > Class Template Reference</code>	1442
5.248.1	Detailed Description	1444
5.248.2	Member Function Documentation	1444
5.248.2.1	<code>_M_detach_all</code>	1444
5.248.2.2	<code>_M_detach_singular</code>	1445
5.248.2.3	<code>_M_get_mutex</code>	1445
5.248.2.4	<code>_M_invalidate_all</code>	1445
5.248.2.5	<code>_M_invalidate_if</code>	1445
5.248.2.6	<code>_M_revalidate_singular</code>	1446
5.248.2.7	<code>_M_swap</code>	1446
5.248.2.8	<code>_M_transfer_iter</code>	1446
5.248.3	Member Data Documentation	1446
5.248.3.1	<code>_M_const_iterators</code>	1446
5.248.3.2	<code>_M_iterators</code>	1446
5.248.3.3	<code>_M_version</code>	1447
5.249	<code>std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference</code>	1448
5.249.1	Detailed Description	1450
5.249.2	Member Function Documentation	1450
5.249.2.1	<code>_M_detach_all</code>	1450
5.249.2.2	<code>_M_detach_singular</code>	1450
5.249.2.3	<code>_M_get_mutex</code>	1450
5.249.2.4	<code>_M_invalidate_all</code>	1451
5.249.2.5	<code>_M_invalidate_if</code>	1451
5.249.2.6	<code>_M_revalidate_singular</code>	1451
5.249.2.7	<code>_M_swap</code>	1451
5.249.2.8	<code>_M_transfer_iter</code>	1452
5.249.3	Member Data Documentation	1452

5.249.3.1	_M_const_iterators	1452
5.249.3.2	_M_iterators	1452
5.249.3.3	_M_version	1452
5.250	std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1453
5.250.1	Detailed Description	1455
5.250.2	Member Function Documentation	1455
5.250.2.1	_M_detach_all	1455
5.250.2.2	_M_detach_singular	1455
5.250.2.3	_M_get_mutex	1455
5.250.2.4	_M_invalidate_all	1455
5.250.2.5	_M_invalidate_if	1456
5.250.2.6	_M_revalidate_singular	1456
5.250.2.7	_M_swap	1456
5.250.2.8	_M_transfer_iter	1456
5.250.3	Member Data Documentation	1457
5.250.3.1	_M_const_iterators	1457
5.250.3.2	_M_iterators	1457
5.250.3.3	_M_version	1457
5.251	std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc > Class Template Reference	1458
5.251.1	Detailed Description	1460
5.251.2	Member Function Documentation	1460
5.251.2.1	_M_detach_all	1460
5.251.2.2	_M_detach_singular	1460
5.251.2.3	_M_get_mutex	1460
5.251.2.4	_M_invalidate_all	1460
5.251.2.5	_M_invalidate_if	1461
5.251.2.6	_M_revalidate_singular	1461
5.251.2.7	_M_swap	1461
5.251.2.8	_M_transfer_iter	1461
5.251.3	Member Data Documentation	1461

5.251.3.1	_M_const_iterators	1461
5.251.3.2	_M_iterators	1462
5.251.3.3	_M_version	1462
5.252	std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1463
5.252.1	Detailed Description	1465
5.252.2	Member Function Documentation	1465
5.252.2.1	_M_detach_all	1465
5.252.2.2	_M_detach_singular	1465
5.252.2.3	_M_get_mutex	1465
5.252.2.4	_M_invalidate_all	1466
5.252.2.5	_M_invalidate_if	1466
5.252.2.6	_M_revalidate_singular	1466
5.252.2.7	_M_swap	1466
5.252.2.8	_M_transfer_iter	1467
5.252.3	Member Data Documentation	1467
5.252.3.1	_M_const_iterators	1467
5.252.3.2	_M_iterators	1467
5.252.3.3	_M_version	1467
5.253	std::__debug::vector< _Tp, _Allocator > Class Template Reference	1468
5.253.1	Detailed Description	1471
5.253.2	Constructor & Destructor Documentation	1471
5.253.2.1	vector	1471
5.253.3	Member Function Documentation	1471
5.253.3.1	_M_detach_all	1471
5.253.3.2	_M_detach_singular	1471
5.253.3.3	_M_get_mutex	1471
5.253.3.4	_M_invalidate_all	1472
5.253.3.5	_M_invalidate_if	1472
5.253.3.6	_M_revalidate_singular	1472
5.253.3.7	_M_swap	1472

5.253.3.8 <code>_M_transfer_iter</code>	1472
5.253.4 Member Data Documentation	1473
5.253.4.1 <code>_M_const_iterators</code>	1473
5.253.4.2 <code>_M_iterators</code>	1473
5.253.4.3 <code>_M_version</code>	1473
5.254 <code>std::__declval_protector< _Tp ></code> Struct Template Reference	1474
5.254.1 Detailed Description	1474
5.255 <code>std::__exception_ptr::exception_ptr</code> Class Reference	1475
5.255.1 Detailed Description	1475
5.256 <code>std::__future_base</code> Struct Reference	1476
5.256.1 Detailed Description	1476
5.257 <code>std::__future_base::Ptr< _Res ></code> Struct Template Reference	1477
5.257.1 Detailed Description	1477
5.258 <code>std::__future_base::Result< _Res ></code> Struct Template Reference	1478
5.258.1 Detailed Description	1478
5.259 <code>std::__future_base::Result< _Res & ></code> Struct Template Reference	1479
5.259.1 Detailed Description	1479
5.260 <code>std::__future_base::Result< void ></code> Struct Template Reference	1480
5.260.1 Detailed Description	1480
5.261 <code>std::__future_base::Result_base</code> Struct Reference	1481
5.261.1 Detailed Description	1481
5.262 <code>std::__future_base::State</code> Class Reference	1482
5.262.1 Detailed Description	1482
5.263 <code>std::__is_location_invariant< _Tp ></code> Struct Template Reference	1483
5.263.1 Detailed Description	1483
5.264 <code>std::__is_member_pointer_helper< _Tp ></code> Struct Template Reference	1484
5.264.1 Detailed Description	1484
5.265 <code>std::__numeric_limits_base</code> Struct Reference	1485
5.265.1 Detailed Description	1486
5.265.2 Member Data Documentation	1486
5.265.2.1 <code>digits</code>	1486

5.265.2.2	digits10	1486
5.265.2.3	has_denorm	1486
5.265.2.4	has_denorm_loss	1486
5.265.2.5	has_infinity	1487
5.265.2.6	has_quiet_NaN	1487
5.265.2.7	has_signaling_NaN	1487
5.265.2.8	is_bounded	1487
5.265.2.9	is_exact	1487
5.265.2.10	is_iec559	1487
5.265.2.11	is_integer	1487
5.265.2.12	is_modulo	1488
5.265.2.13	is_signed	1488
5.265.2.14	is_specialized	1488
5.265.2.15	max_digits10	1488
5.265.2.16	max_exponent	1488
5.265.2.17	max_exponent10	1488
5.265.2.18	min_exponent	1489
5.265.2.19	min_exponent10	1489
5.265.2.20	radix	1489
5.265.2.21	round_style	1489
5.265.2.22	traps_before	1489
5.265.2.23	traps	1489
5.266	std::__parallel::__CRandNumber<_MustBeInt> Struct Template Reference	1490
5.266.1	Detailed Description	1490
5.267	std::__profile::bitset<_Nb> Class Template Reference	1491
5.267.1	Detailed Description	1492
5.268	std::__profile::deque<_Tp, _Allocator> Class Template Reference	1493
5.268.1	Detailed Description	1495
5.269	std::__profile::list<_Tp, _Allocator> Class Template Reference	1496
5.269.1	Detailed Description	1498

5.270	<code>std::__profile::map< _Key, _Tp, _Compare, _Allocator ></code> Class Template Reference	1499
5.270.1	Detailed Description	1500
5.271	<code>std::__profile::multimap< _Key, _Tp, _Compare, _Allocator ></code> Class Template Reference	1502
5.271.1	Detailed Description	1503
5.272	<code>std::__profile::multiset< _Key, _Compare, _Allocator ></code> Class Template Reference	1505
5.272.1	Detailed Description	1506
5.273	<code>std::__profile::set< _Key, _Compare, _Allocator ></code> Class Template Reference	1507
5.273.1	Detailed Description	1508
5.274	<code>std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	1509
5.274.1	Detailed Description	1510
5.275	<code>std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	1511
5.275.1	Detailed Description	1512
5.276	<code>std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc ></code> Class Template Reference	1513
5.276.1	Detailed Description	1514
5.277	<code>std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc ></code> Class Template Reference	1515
5.277.1	Detailed Description	1516
5.278	<code>std::_Base_bitset< _Nw ></code> Struct Template Reference	1517
5.278.1	Detailed Description	1518
5.278.2	Member Data Documentation	1518
5.278.2.1	<code>_M_w</code>	1518
5.279	<code>std::_Base_bitset< 0 ></code> Struct Template Reference	1519
5.279.1	Detailed Description	1520
5.280	<code>std::_Base_bitset< 1 ></code> Struct Template Reference	1521
5.280.1	Detailed Description	1522
5.281	<code>std::_Build_index_tuple< _Num ></code> Struct Template Reference	1523
5.281.1	Detailed Description	1523

5.282std::_Deque_base< _Tp, _Alloc > Class Template Reference	1524
5.282.1 Detailed Description	1525
5.282.2 Member Function Documentation	1525
5.282.2.1 _M_initialize_map	1525
5.283std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference .	1527
5.283.1 Detailed Description	1528
5.283.2 Member Function Documentation	1528
5.283.2.1 _M_set_node	1528
5.284std::_Derives_from_binary_function< _Tp > Struct Template Reference	1529
5.284.1 Detailed Description	1529
5.285std::_Derives_from_unary_function< _Tp > Struct Template Reference	1530
5.285.1 Detailed Description	1530
5.286std::_Function_base Class Reference	1531
5.286.1 Detailed Description	1531
5.287std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct Template Reference	1533
5.287.1 Detailed Description	1533
5.288std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference . . .	1534
5.288.1 Detailed Description	1535
5.289std::_Fwd_list_const_iterator< _Tp, _Alloc > Struct Template Refer- ence	1536
5.289.1 Detailed Description	1536
5.290std::_Fwd_list_iterator< _Tp, _Alloc > Struct Template Reference . .	1538
5.290.1 Detailed Description	1538
5.291std::_Fwd_list_node< _Tp, _Alloc > Struct Template Reference . . .	1539
5.291.1 Detailed Description	1540
5.292std::_Fwd_list_node_base< _Alloc > Struct Template Reference . . .	1541
5.292.1 Detailed Description	1542
5.293std::_Has_result_type_helper< _Tp > Class Template Reference . . .	1543
5.293.1 Detailed Description	1543
5.294std::_Index_tuple< _Indexes > Struct Template Reference	1544
5.294.1 Detailed Description	1544

5.295	<code>std::_List_base< _Tp, _Alloc ></code> Class Template Reference	1545
5.295.1	Detailed Description	1546
5.296	<code>std::_List_const_iterator< _Tp ></code> Struct Template Reference	1547
5.296.1	Detailed Description	1547
5.297	<code>std::_List_iterator< _Tp ></code> Struct Template Reference	1549
5.297.1	Detailed Description	1549
5.298	<code>std::_List_node< _Tp ></code> Struct Template Reference	1550
5.298.1	Detailed Description	1550
5.298.2	Member Data Documentation	1551
5.298.2.1	<code>_M_data</code>	1551
5.299	<code>std::_List_node_base</code> Struct Reference	1552
5.299.1	Detailed Description	1552
5.300	<code>std::_Maybe_get_result_type< _Has_result_type, _Functor ></code> Struct Template Reference	1553
5.300.1	Detailed Description	1553
5.301	<code>std::_Maybe_unary_or_binary_function< _Res, _ArgTypes ></code> Struct Template Reference	1554
5.301.1	Detailed Description	1554
5.302	<code>std::_Maybe_unary_or_binary_function< _Res, _T1 ></code> Struct Template Reference	1555
5.302.1	Detailed Description	1555
5.302.2	Member Typedef Documentation	1555
5.302.2.1	<code>argument_type</code>	1555
5.302.2.2	<code>result_type</code>	1556
5.303	<code>std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 ></code> Struct Template Reference	1557
5.303.1	Detailed Description	1557
5.303.2	Member Typedef Documentation	1558
5.303.2.1	<code>first_argument_type</code>	1558
5.303.2.2	<code>result_type</code>	1558
5.303.2.3	<code>second_argument_type</code>	1558
5.304	<code>std::_Maybe_wrap_member_pointer< _Tp ></code> Struct Template Reference	1559

5.304.1 Detailed Description	1559
5.305std::_Maybe_wrap_member_pointer< _Tp _Class::* > Struct Template Reference	1560
5.305.1 Detailed Description	1560
5.306std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const > Class Template Reference	1561
5.306.1 Detailed Description	1561
5.307std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class Template Reference	1563
5.307.1 Detailed Description	1563
5.308std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile > Class Template Reference	1564
5.308.1 Detailed Description	1564
5.309std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) > Class Template Reference	1566
5.309.1 Detailed Description	1566
5.310std::_Mu< _Arg, false, false > Class Template Reference	1568
5.310.1 Detailed Description	1568
5.311std::_Mu< _Arg, false, true > Class Template Reference	1569
5.311.1 Detailed Description	1569
5.312std::_Mu< _Arg, true, false > Class Template Reference	1570
5.312.1 Detailed Description	1570
5.313std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference	1571
5.313.1 Detailed Description	1571
5.314std::_Placeholder< _Num > Struct Template Reference	1572
5.314.1 Detailed Description	1572
5.315std::_Reference_wrapper_base< _Tp > Struct Template Reference	1573
5.315.1 Detailed Description	1573
5.316std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference	1574
5.316.1 Detailed Description	1574
5.317std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference	1575

5.317.1 Detailed Description	1575
5.318std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference	1576
5.318.1 Detailed Description	1576
5.319std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference	1577
5.319.1 Detailed Description	1578
5.319.2 Constructor & Destructor Documentation	1578
5.319.2.1 _Temporary_buffer	1578
5.319.3 Member Function Documentation	1578
5.319.3.1 begin	1578
5.319.3.2 end	1578
5.319.3.3 requested_size	1579
5.319.3.4 size	1579
5.320std::_Tuple_impl< _Idx > Struct Template Reference	1580
5.320.1 Detailed Description	1580
5.321std::_Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference	1581
5.321.1 Detailed Description	1582
5.322std::_Vector_base< _Tp, _Alloc > Struct Template Reference	1583
5.322.1 Detailed Description	1583
5.323std::_Weak_result_type< _Functor > Struct Template Reference	1585
5.323.1 Detailed Description	1585
5.324std::_Weak_result_type_impl< _Functor > Struct Template Reference	1586
5.324.1 Detailed Description	1586
5.325std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Template Reference	1587
5.325.1 Detailed Description	1587
5.326std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)> Struct Template Reference	1588
5.326.1 Detailed Description	1588
5.327std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference	1589
5.327.1 Detailed Description	1589

5.328	<code>std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const</code> <code>> Struct Template Reference</code>	1590
5.328.1	Detailed Description	1590
5.329	<code>std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const</code> <code>volatile > Struct Template Reference</code>	1591
5.329.1	Detailed Description	1591
5.330	<code>std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)</code> <code>volatile > Struct Template Reference</code>	1592
5.330.1	Detailed Description	1592
5.331	<code>std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)></code> <code>Struct Template Reference</code>	1593
5.331.1	Detailed Description	1593
5.332	<code>std::add_const< _Tp > Struct Template Reference</code>	1594
5.332.1	Detailed Description	1594
5.333	<code>std::add_cv< _Tp > Struct Template Reference</code>	1595
5.333.1	Detailed Description	1595
5.334	<code>std::add_lvalue_reference< _Tp > Struct Template Reference</code>	1596
5.334.1	Detailed Description	1596
5.335	<code>std::add_pointer< _Tp > Struct Template Reference</code>	1597
5.335.1	Detailed Description	1597
5.336	<code>std::add_rvalue_reference< _Tp > Struct Template Reference</code>	1598
5.336.1	Detailed Description	1598
5.337	<code>std::add_volatile< _Tp > Struct Template Reference</code>	1599
5.337.1	Detailed Description	1599
5.338	<code>std::adopt_lock_t Struct Reference</code>	1600
5.338.1	Detailed Description	1600
5.339	<code>std::aligned_storage< _Len, _Align > Struct Template Reference</code>	1601
5.339.1	Detailed Description	1601
5.340	<code>std::alignment_of< _Tp > Struct Template Reference</code>	1602
5.340.1	Detailed Description	1602
5.341	<code>std::allocator< _Tp > Class Template Reference</code>	1603
5.341.1	Detailed Description	1604

5.342	<code>std::allocator< void ></code> Class Template Reference	1605
5.342.1	Detailed Description	1605
5.343	<code>std::array< _Tp, _Nm ></code> Struct Template Reference	1606
5.343.1	Detailed Description	1607
5.344	<code>std::atomic< _Tp ></code> Struct Template Reference	1608
5.344.1	Detailed Description	1608
5.345	<code>std::atomic< _Tp * ></code> Struct Template Reference	1609
5.345.1	Detailed Description	1609
5.346	<code>std::atomic< bool ></code> Struct Template Reference	1610
5.346.1	Detailed Description	1610
5.347	<code>std::atomic< char ></code> Struct Template Reference	1611
5.347.1	Detailed Description	1611
5.348	<code>std::atomic< char16_t ></code> Struct Template Reference	1612
5.348.1	Detailed Description	1612
5.349	<code>std::atomic< char32_t ></code> Struct Template Reference	1613
5.349.1	Detailed Description	1613
5.350	<code>std::atomic< int ></code> Struct Template Reference	1614
5.350.1	Detailed Description	1614
5.351	<code>std::atomic< long ></code> Struct Template Reference	1615
5.351.1	Detailed Description	1615
5.352	<code>std::atomic< long long ></code> Struct Template Reference	1616
5.352.1	Detailed Description	1616
5.353	<code>std::atomic< short ></code> Struct Template Reference	1617
5.353.1	Detailed Description	1617
5.354	<code>std::atomic< signed char ></code> Struct Template Reference	1618
5.354.1	Detailed Description	1618
5.355	<code>std::atomic< unsigned char ></code> Struct Template Reference	1619
5.355.1	Detailed Description	1619
5.356	<code>std::atomic< unsigned int ></code> Struct Template Reference	1620
5.356.1	Detailed Description	1620
5.357	<code>std::atomic< unsigned long ></code> Struct Template Reference	1621

5.357.1 Detailed Description	1621
5.358std::atomic< unsigned long long > Struct Template Reference	1622
5.358.1 Detailed Description	1622
5.359std::atomic< unsigned short > Struct Template Reference	1623
5.359.1 Detailed Description	1623
5.360std::atomic< void * > Struct Template Reference	1624
5.360.1 Detailed Description	1624
5.361std::atomic< wchar_t > Struct Template Reference	1625
5.361.1 Detailed Description	1625
5.362std::auto_ptr< _Tp > Class Template Reference	1626
5.362.1 Detailed Description	1626
5.362.2 Member Typedef Documentation	1627
5.362.2.1 element_type	1627
5.362.3 Constructor & Destructor Documentation	1627
5.362.3.1 auto_ptr	1627
5.362.3.2 auto_ptr	1627
5.362.3.3 auto_ptr	1628
5.362.3.4 ~auto_ptr	1628
5.362.3.5 auto_ptr	1628
5.362.4 Member Function Documentation	1629
5.362.4.1 get	1629
5.362.4.2 operator*	1629
5.362.4.3 operator->	1629
5.362.4.4 operator=	1629
5.362.4.5 operator=	1630
5.362.4.6 release	1630
5.362.4.7 reset	1630
5.363std::auto_ptr_ref< _Tp1 > Struct Template Reference	1632
5.363.1 Detailed Description	1632
5.364std::back_insert_iterator< _Container > Class Template Reference	1633
5.364.1 Detailed Description	1634

5.364.2 Member Typedef Documentation	1634
5.364.2.1 container_type	1634
5.364.2.2 difference_type	1634
5.364.2.3 iterator_category	1634
5.364.2.4 pointer	1634
5.364.2.5 reference	1635
5.364.2.6 value_type	1635
5.364.3 Constructor & Destructor Documentation	1635
5.364.3.1 back_insert_iterator	1635
5.364.4 Member Function Documentation	1635
5.364.4.1 operator*	1635
5.364.4.2 operator++	1635
5.364.4.3 operator++	1636
5.364.4.4 operator=	1636
5.365std::bad_alloc Class Reference	1637
5.365.1 Detailed Description	1637
5.365.2 Member Function Documentation	1637
5.365.2.1 what	1637
5.366std::bad_cast Class Reference	1638
5.366.1 Detailed Description	1638
5.366.2 Member Function Documentation	1638
5.366.2.1 what	1638
5.367std::bad_exception Class Reference	1639
5.367.1 Detailed Description	1639
5.367.2 Member Function Documentation	1639
5.367.2.1 what	1639
5.368std::bad_function_call Class Reference	1640
5.368.1 Detailed Description	1640
5.368.2 Member Function Documentation	1640
5.368.2.1 what	1640
5.369std::bad_typeid Class Reference	1641

5.369.1 Detailed Description	1641
5.369.2 Member Function Documentation	1641
5.369.2.1 what	1641
5.370std::bad_weak_ptr Class Reference	1642
5.370.1 Detailed Description	1642
5.370.2 Member Function Documentation	1642
5.370.2.1 what	1642
5.371std::basic_filebuf< _CharT, _Traits > Class Template Reference	1643
5.371.1 Detailed Description	1646
5.371.2 Member Typedef Documentation	1646
5.371.2.1 __streambuf_type	1646
5.371.2.2 char_type	1647
5.371.2.3 int_type	1647
5.371.2.4 off_type	1647
5.371.2.5 pos_type	1647
5.371.2.6 traits_type	1648
5.371.3 Constructor & Destructor Documentation	1648
5.371.3.1 basic_filebuf	1648
5.371.3.2 ~basic_filebuf	1648
5.371.4 Member Function Documentation	1648
5.371.4.1 _M_create_pback	1648
5.371.4.2 _M_destroy_pback	1649
5.371.4.3 _M_set_buffer	1649
5.371.4.4 close	1649
5.371.4.5 eback	1650
5.371.4.6 egptr	1650
5.371.4.7 epptr	1651
5.371.4.8 gbump	1651
5.371.4.9 getloc	1651
5.371.4.10gptr	1652
5.371.4.11imbue	1652

5.371.4.12	n_avail	1653
5.371.4.13	s_open	1653
5.371.4.14	open	1653
5.371.4.15	open	1654
5.371.4.16	overflow	1655
5.371.4.17	backfail	1655
5.371.4.18	base	1656
5.371.4.19	bump	1657
5.371.4.20	ptr	1657
5.371.4.21	pubimbue	1657
5.371.4.22	pubseekoff	1658
5.371.4.23	pubseekpos	1658
5.371.4.24	pubsetbuf	1658
5.371.4.25	pubsync	1659
5.371.4.26	sbumpc	1659
5.371.4.27	seekoff	1659
5.371.4.28	seekpos	1660
5.371.4.29	setbuf	1660
5.371.4.30	setg	1661
5.371.4.31	setp	1661
5.371.4.32	getc	1662
5.371.4.33	getn	1662
5.371.4.34	showmanyc	1662
5.371.4.35	nextc	1663
5.371.4.36	putbackc	1664
5.371.4.37	putc	1664
5.371.4.38	putn	1665
5.371.4.39	stossc	1665
5.371.4.40	sungetc	1665
5.371.4.41	sync	1666
5.371.4.42	unflow	1666

5.371.4.43	underflow	1666
5.371.4.44	xsgetn	1667
5.371.4.45	xspn	1668
5.371.5	Member Data Documentation	1669
5.371.5.1	_M_buf	1669
5.371.5.2	_M_buf_locale	1669
5.371.5.3	_M_buf_size	1669
5.371.5.4	_M_ext_buf	1669
5.371.5.5	_M_ext_buf_size	1670
5.371.5.6	_M_ext_next	1670
5.371.5.7	_M_in_beg	1670
5.371.5.8	_M_in_cur	1670
5.371.5.9	_M_in_end	1671
5.371.5.10	_M_mode	1671
5.371.5.11	_M_out_beg	1671
5.371.5.12	_M_out_cur	1672
5.371.5.13	_M_out_end	1672
5.371.5.14	_M_pback	1672
5.371.5.15	_M_pback_cur_save	1672
5.371.5.16	_M_pback_end_save	1673
5.371.5.17	_M_pback_init	1673
5.371.5.18	_M_reading	1673
5.372	std::basic_fstream<_CharT, _Traits> Class Template Reference	1675
5.372.1	Detailed Description	1682
5.372.2	Member Typedef Documentation	1683
5.372.2.1	__ctype_type	1683
5.372.2.2	__ctype_type	1683
5.372.2.3	__num_get_type	1683
5.372.2.4	__num_put_type	1683
5.372.2.5	__num_put_type	1684
5.372.2.6	char_type	1684

5.372.2.7	event_callback	1684
5.372.2.8	fmtflags	1684
5.372.2.9	int_type	1685
5.372.2.10	iostate	1686
5.372.2.11	loff_type	1686
5.372.2.12	openmode	1686
5.372.2.13	pos_type	1687
5.372.2.14	seekdir	1687
5.372.2.15	traits_type	1687
5.372.3	Member Enumeration Documentation	1687
5.372.3.1	event	1687
5.372.4	Constructor & Destructor Documentation	1688
5.372.4.1	basic_fstream	1688
5.372.4.2	basic_fstream	1688
5.372.4.3	basic_fstream	1688
5.372.4.4	~basic_fstream	1689
5.372.5	Member Function Documentation	1689
5.372.5.1	_M_getloc	1689
5.372.5.2	_M_write	1689
5.372.5.3	bad	1690
5.372.5.4	clear	1690
5.372.5.5	close	1690
5.372.5.6	copyfmt	1691
5.372.5.7	eof	1691
5.372.5.8	exceptions	1691
5.372.5.9	exceptions	1692
5.372.5.10	fail	1692
5.372.5.11	fill	1693
5.372.5.12	fill	1693
5.372.5.13	flags	1694
5.372.5.14	flags	1694

5.372.5.15flush	1694
5.372.5.16gcount	1695
5.372.5.17get	1695
5.372.5.18get	1695
5.372.5.19get	1696
5.372.5.20get	1697
5.372.5.21get	1697
5.372.5.22get	1698
5.372.5.23getline	1698
5.372.5.24getline	1699
5.372.5.25getloc	1700
5.372.5.26good	1700
5.372.5.27ignore	1700
5.372.5.28ignore	1701
5.372.5.29ignore	1701
5.372.5.30mbue	1702
5.372.5.31imit	1703
5.372.5.32is_open	1703
5.372.5.33iword	1703
5.372.5.34narrow	1704
5.372.5.35open	1704
5.372.5.36open	1704
5.372.5.37operator void *	1705
5.372.5.38operator!	1705
5.372.5.39operator<<	1705
5.372.5.40operator<<	1706
5.372.5.41operator<<	1707
5.372.5.42operator<<	1707
5.372.5.43operator<<	1708
5.372.5.44operator<<	1708
5.372.5.45operator<<	1709

5.372.5.46operator<<	1710
5.372.5.47operator<<	1710
5.372.5.48operator<<	1711
5.372.5.49operator<<	1711
5.372.5.50operator<<	1712
5.372.5.51operator<<	1713
5.372.5.52operator<<	1713
5.372.5.53operator<<	1714
5.372.5.54operator<<	1714
5.372.5.55operator<<	1714
5.372.5.56operator>>	1714
5.372.5.57operator>>	1715
5.372.5.58operator>>	1716
5.372.5.59operator>>	1716
5.372.5.60operator>>	1717
5.372.5.61operator>>	1718
5.372.5.62operator>>	1718
5.372.5.63operator>>	1719
5.372.5.64operator>>	1719
5.372.5.65operator>>	1720
5.372.5.66operator>>	1721
5.372.5.67operator>>	1721
5.372.5.68operator>>	1722
5.372.5.69operator>>	1722
5.372.5.70operator>>	1723
5.372.5.71operator>>	1723
5.372.5.72operator>>	1723
5.372.5.73peek	1724
5.372.5.74precision	1724
5.372.5.75precision	1724
5.372.5.76put	1725

5.372.5.77	putback	1725
5.372.5.78	ppword	1726
5.372.5.79	rdbuf	1726
5.372.5.80	rdbuf	1727
5.372.5.81	rdstate	1727
5.372.5.82	read	1727
5.372.5.83	readsome	1728
5.372.5.84	register_callback	1729
5.372.5.85	seekg	1729
5.372.5.86	seekg	1730
5.372.5.87	seekp	1730
5.372.5.88	seekp	1731
5.372.5.89	setf	1731
5.372.5.90	setf	1732
5.372.5.91	lsetstate	1732
5.372.5.92	sync	1733
5.372.5.93	sync_with_stdio	1733
5.372.5.94	tellg	1734
5.372.5.95	tellp	1734
5.372.5.96	tie	1734
5.372.5.97	tie	1735
5.372.5.98	unset	1735
5.372.5.99	unsetf	1736
5.372.5.100	iden	1736
5.372.5.101	width	1737
5.372.5.102	width	1737
5.372.5.103	write	1737
5.372.5.104	walloc	1738
5.372.6	Member Data Documentation	1738
5.372.6.1	_M_gcount	1738
5.372.6.2	adjustfield	1739

5.372.6.3	app	1739
5.372.6.4	ate	1739
5.372.6.5	badbit	1739
5.372.6.6	basefield	1740
5.372.6.7	beg	1740
5.372.6.8	binary	1740
5.372.6.9	boolalpha	1740
5.372.6.10	cur	1740
5.372.6.11	dec	1741
5.372.6.12	end	1741
5.372.6.13	eofbit	1741
5.372.6.14	failbit	1741
5.372.6.15	fixed	1742
5.372.6.16	floatfield	1742
5.372.6.17	goodbit	1742
5.372.6.18	hex	1742
5.372.6.19	n	1742
5.372.6.20	internal	1743
5.372.6.21	left	1743
5.372.6.22	oct	1743
5.372.6.23	out	1743
5.372.6.24	right	1744
5.372.6.25	scientific	1744
5.372.6.26	showbase	1744
5.372.6.27	showpoint	1744
5.372.6.28	showpos	1744
5.372.6.29	skipws	1744
5.372.6.30	trunc	1744
5.372.6.31	unitbuf	1745
5.372.6.32	uppercase	1745
5.373	std::basic_ifstream<_CharT, _Traits> Class Template Reference	1746

5.373.1 Detailed Description	1752
5.373.2 Member Typedef Documentation	1752
5.373.2.1 __ctype_type	1752
5.373.2.2 __num_get_type	1752
5.373.2.3 __num_put_type	1753
5.373.2.4 char_type	1753
5.373.2.5 event_callback	1753
5.373.2.6 fmtflags	1753
5.373.2.7 int_type	1754
5.373.2.8 iostate	1755
5.373.2.9 off_type	1755
5.373.2.10openmode	1755
5.373.2.11pos_type	1756
5.373.2.12seekdir	1756
5.373.2.13traits_type	1756
5.373.3 Member Enumeration Documentation	1756
5.373.3.1 event	1756
5.373.4 Constructor & Destructor Documentation	1757
5.373.4.1 basic_ifstream	1757
5.373.4.2 basic_ifstream	1757
5.373.4.3 basic_ifstream	1757
5.373.4.4 ~basic_ifstream	1758
5.373.5 Member Function Documentation	1758
5.373.5.1 _M_getloc	1758
5.373.5.2 bad	1758
5.373.5.3 clear	1759
5.373.5.4 close	1759
5.373.5.5 copyfmt	1759
5.373.5.6 eof	1760
5.373.5.7 exceptions	1760
5.373.5.8 exceptions	1761

5.373.5.9 fail	1761
5.373.5.10fill	1761
5.373.5.11fill	1762
5.373.5.12flags	1762
5.373.5.13flags	1762
5.373.5.14gcount	1763
5.373.5.15get	1763
5.373.5.16get	1763
5.373.5.17get	1764
5.373.5.18get	1765
5.373.5.19get	1765
5.373.5.20get	1766
5.373.5.21getline	1766
5.373.5.22getline	1767
5.373.5.23getloc	1768
5.373.5.24good	1768
5.373.5.25ignore	1768
5.373.5.26ignore	1769
5.373.5.27ignore	1769
5.373.5.28imbue	1770
5.373.5.29init	1771
5.373.5.30is_open	1771
5.373.5.31iword	1771
5.373.5.32narrow	1772
5.373.5.33open	1772
5.373.5.34open	1772
5.373.5.35operator void *	1773
5.373.5.36operator!	1773
5.373.5.37operator>>	1773
5.373.5.38operator>>	1774
5.373.5.39operator>>	1775

5.373.5.40operator>>	1775
5.373.5.41operator>>	1776
5.373.5.42operator>>	1776
5.373.5.43operator>>	1777
5.373.5.44operator>>	1778
5.373.5.45operator>>	1778
5.373.5.46operator>>	1779
5.373.5.47operator>>	1779
5.373.5.48operator>>	1780
5.373.5.49operator>>	1781
5.373.5.50operator>>	1781
5.373.5.51operator>>	1782
5.373.5.52operator>>	1782
5.373.5.53operator>>	1782
5.373.5.54peek	1782
5.373.5.55precision	1783
5.373.5.56precision	1783
5.373.5.57putback	1783
5.373.5.58pword	1784
5.373.5.59rdbuf	1784
5.373.5.60rdbuf	1785
5.373.5.61rdstate	1785
5.373.5.62read	1786
5.373.5.63readsome	1786
5.373.5.64register_callback	1787
5.373.5.65seekg	1787
5.373.5.66seekg	1788
5.373.5.67setf	1788
5.373.5.68setf	1789
5.373.5.69setstate	1789
5.373.5.70sync	1790

5.373.5.7	l	sync_with_stdio	1790
5.373.5.7	2	tellg	1791
5.373.5.7	3	tie	1791
5.373.5.7	4	tie	1792
5.373.5.7	5	unset	1792
5.373.5.7	6	unsetf	1793
5.373.5.7	7	widen	1793
5.373.5.7	8	width	1793
5.373.5.7	9	width	1794
5.373.5.8	0	kalloc	1794
5.373.6		Member Data Documentation	1794
5.373.6.1		_M_gcount	1794
5.373.6.2		adjustfield	1795
5.373.6.3		app	1795
5.373.6.4		ate	1795
5.373.6.5		badbit	1795
5.373.6.6		basefield	1796
5.373.6.7		beg	1796
5.373.6.8		binary	1796
5.373.6.9		boolalpha	1796
5.373.6.10		cur	1796
5.373.6.1		dec	1797
5.373.6.12		end	1797
5.373.6.13		eofbit	1797
5.373.6.14		failbit	1797
5.373.6.15		fixed	1798
5.373.6.16		floatfield	1798
5.373.6.17		goodbit	1798
5.373.6.18		hex	1798
5.373.6.19		n	1799
5.373.6.20		internal	1799

5.373.6.21left	1799
5.373.6.22oct	1799
5.373.6.23out	1799
5.373.6.24right	1800
5.373.6.25scientific	1800
5.373.6.26showbase	1800
5.373.6.27showpoint	1800
5.373.6.28showpos	1800
5.373.6.29skipws	1800
5.373.6.30trunc	1800
5.373.6.31unitbuf	1801
5.373.6.32uppercase	1801
5.374std::basic_ios<_CharT, _Traits> Class Template Reference	1802
5.374.1 Detailed Description	1806
5.374.2 Member Typedef Documentation	1806
5.374.2.1 __ctype_type	1806
5.374.2.2 __num_get_type	1806
5.374.2.3 __num_put_type	1806
5.374.2.4 char_type	1807
5.374.2.5 event_callback	1807
5.374.2.6 fmtflags	1807
5.374.2.7 int_type	1808
5.374.2.8 iostate	1809
5.374.2.9 off_type	1809
5.374.2.10openmode	1809
5.374.2.11pos_type	1810
5.374.2.12seekdir	1810
5.374.2.13traits_type	1810
5.374.3 Member Enumeration Documentation	1811
5.374.3.1 event	1811
5.374.4 Constructor & Destructor Documentation	1811

5.374.4.1	<code>basic_ios</code>	1811
5.374.4.2	<code>~basic_ios</code>	1811
5.374.4.3	<code>basic_ios</code>	1811
5.374.5	Member Function Documentation	1811
5.374.5.1	<code>_M_getloc</code>	1811
5.374.5.2	<code>bad</code>	1812
5.374.5.3	<code>clear</code>	1812
5.374.5.4	<code>copyfmt</code>	1813
5.374.5.5	<code>eof</code>	1813
5.374.5.6	exceptions	1813
5.374.5.7	exceptions	1814
5.374.5.8	<code>fail</code>	1814
5.374.5.9	<code>fill</code>	1815
5.374.5.10	<code>fill</code>	1815
5.374.5.11	<code>iflags</code>	1816
5.374.5.12	<code>iflags</code>	1816
5.374.5.13	<code>getloc</code>	1816
5.374.5.14	<code>good</code>	1817
5.374.5.15	<code>imbue</code>	1817
5.374.5.16	<code>init</code>	1817
5.374.5.17	<code>word</code>	1818
5.374.5.18	<code>narrow</code>	1818
5.374.5.19	operator <code>void*</code>	1819
5.374.5.20	operator <code>!</code>	1819
5.374.5.21	<code>precision</code>	1819
5.374.5.22	<code>precision</code>	1819
5.374.5.23	<code>word</code>	1820
5.374.5.24	<code>rdbuf</code>	1820
5.374.5.25	<code>rdbuf</code>	1821
5.374.5.26	<code>rdstate</code>	1821
5.374.5.27	<code>register_callback</code>	1822

5.374.5.28setf	1822
5.374.5.29setf	1822
5.374.5.30setstate	1823
5.374.5.31sync_with_stdio	1823
5.374.5.32ie	1824
5.374.5.33ie	1824
5.374.5.34unsetf	1824
5.374.5.35widen	1825
5.374.5.36width	1825
5.374.5.37width	1826
5.374.5.38xalloc	1826
5.374.6 Member Data Documentation	1826
5.374.6.1 adjustfield	1826
5.374.6.2 app	1826
5.374.6.3 ate	1827
5.374.6.4 badbit	1827
5.374.6.5 basefield	1827
5.374.6.6 beg	1827
5.374.6.7 binary	1828
5.374.6.8 boolalpha	1828
5.374.6.9 cur	1828
5.374.6.10dec	1828
5.374.6.11lend	1828
5.374.6.12eofbit	1829
5.374.6.13failbit	1829
5.374.6.14fixed	1829
5.374.6.15floatfield	1829
5.374.6.16goodbit	1830
5.374.6.17hex	1830
5.374.6.18n	1830
5.374.6.19internal	1830

5.374.6.20left	1831
5.374.6.21oct	1831
5.374.6.22out	1831
5.374.6.23right	1831
5.374.6.24scientific	1831
5.374.6.25showbase	1832
5.374.6.26showpoint	1832
5.374.6.27showpos	1832
5.374.6.28skipws	1832
5.374.6.29trunc	1832
5.374.6.30unitbuf	1832
5.374.6.31uppercase	1832
5.375std::basic_ostream<_CharT,_Traits> Class Template Reference	1834
5.375.1 Detailed Description	1841
5.375.2 Member Typedef Documentation	1841
5.375.2.1 __ctype_type	1841
5.375.2.2 __ctype_type	1841
5.375.2.3 __num_get_type	1841
5.375.2.4 __num_put_type	1842
5.375.2.5 __num_put_type	1842
5.375.2.6 char_type	1842
5.375.2.7 event_callback	1842
5.375.2.8 fmtflags	1843
5.375.2.9 int_type	1844
5.375.2.10ostate	1844
5.375.2.11loff_type	1844
5.375.2.12openmode	1844
5.375.2.13pos_type	1845
5.375.2.14seekdir	1845
5.375.2.15traits_type	1845
5.375.3 Member Enumeration Documentation	1846

5.375.3.1 event	1846
5.375.4 Constructor & Destructor Documentation	1846
5.375.4.1 basic_ostream	1846
5.375.4.2 ~basic_ostream	1846
5.375.5 Member Function Documentation	1846
5.375.5.1 _M_getloc	1846
5.375.5.2 _M_write	1847
5.375.5.3 bad	1847
5.375.5.4 clear	1847
5.375.5.5 copyfmt	1848
5.375.5.6 eof	1848
5.375.5.7 exceptions	1849
5.375.5.8 exceptions	1849
5.375.5.9 fail	1850
5.375.5.10fill	1850
5.375.5.11fill	1851
5.375.5.12flags	1851
5.375.5.13flags	1851
5.375.5.14flush	1852
5.375.5.15gcount	1852
5.375.5.16get	1852
5.375.5.17get	1853
5.375.5.18get	1853
5.375.5.19get	1854
5.375.5.20get	1855
5.375.5.21get	1855
5.375.5.22getline	1856
5.375.5.23getline	1856
5.375.5.24getloc	1857
5.375.5.25good	1857
5.375.5.26ignore	1858

5.375.5.27ignore	1858
5.375.5.28ignore	1859
5.375.5.29imbue	1859
5.375.5.30init	1860
5.375.5.31iword	1860
5.375.5.32narrow	1861
5.375.5.33operator void *	1861
5.375.5.34operator!	1861
5.375.5.35operator<<	1862
5.375.5.36operator<<	1862
5.375.5.37operator<<	1863
5.375.5.38operator<<	1864
5.375.5.39operator<<	1864
5.375.5.40operator<<	1865
5.375.5.41operator<<	1865
5.375.5.42operator<<	1866
5.375.5.43operator<<	1867
5.375.5.44operator<<	1867
5.375.5.45operator<<	1868
5.375.5.46operator<<	1868
5.375.5.47operator<<	1869
5.375.5.48operator<<	1870
5.375.5.49operator<<	1870
5.375.5.50operator<<	1870
5.375.5.51operator<<	1871
5.375.5.52operator>>	1871
5.375.5.53operator>>	1872
5.375.5.54operator>>	1872
5.375.5.55operator>>	1873
5.375.5.56operator>>	1873
5.375.5.57operator>>	1874

5.375.5.58operator>>	1875
5.375.5.59operator>>	1875
5.375.5.60operator>>	1876
5.375.5.61operator>>	1876
5.375.5.62operator>>	1877
5.375.5.63operator>>	1878
5.375.5.64operator>>	1878
5.375.5.65operator>>	1879
5.375.5.66operator>>	1879
5.375.5.67operator>>	1879
5.375.5.68operator>>	1880
5.375.5.69peek	1880
5.375.5.70precision	1880
5.375.5.71precision	1881
5.375.5.72put	1881
5.375.5.73putback	1881
5.375.5.74pword	1882
5.375.5.75rdbuf	1883
5.375.5.76rdbuf	1883
5.375.5.77rdstate	1884
5.375.5.78read	1884
5.375.5.79readsome	1885
5.375.5.80register_callback	1886
5.375.5.81seekg	1886
5.375.5.82seekg	1886
5.375.5.83seekp	1887
5.375.5.84seekp	1888
5.375.5.85setf	1888
5.375.5.86setf	1888
5.375.5.87setstate	1889
5.375.5.88sync	1889

5.375.5.8	sync_with_stdio	1890
5.375.5.9	tellg	1890
5.375.5.9	ltellp	1891
5.375.5.9	tie	1891
5.375.5.9	tie	1892
5.375.5.9	unget	1892
5.375.5.9	unsetf	1893
5.375.5.9	widen	1893
5.375.5.9	width	1893
5.375.5.9	width	1894
5.375.5.9	write	1894
5.375.5.10	alloc	1895
5.375.6	Member Data Documentation	1895
5.375.6.1	_M_gcount	1895
5.375.6.2	adjustfield	1895
5.375.6.3	app	1896
5.375.6.4	ate	1896
5.375.6.5	badbit	1896
5.375.6.6	basefield	1896
5.375.6.7	beg	1896
5.375.6.8	binary	1897
5.375.6.9	boolalpha	1897
5.375.6.10	cur	1897
5.375.6.11	ldec	1897
5.375.6.12	end	1897
5.375.6.13	eofbit	1898
5.375.6.14	failbit	1898
5.375.6.15	fixed	1898
5.375.6.16	floatfield	1898
5.375.6.17	goodbit	1899
5.375.6.18	hex	1899

5.375.6.19n	1899
5.375.6.20internal	1899
5.375.6.21left	1900
5.375.6.22oct	1900
5.375.6.23out	1900
5.375.6.24right	1900
5.375.6.25scientific	1900
5.375.6.26showbase	1901
5.375.6.27showpoint	1901
5.375.6.28showpos	1901
5.375.6.29skipws	1901
5.375.6.30trunc	1901
5.375.6.31unitbuf	1901
5.375.6.32uppercase	1901
5.376std::basic_istream<_CharT, _Traits> Class Template Reference	1903
5.376.1 Detailed Description	1909
5.376.2 Member Typedef Documentation	1909
5.376.2.1 __ctype_type	1909
5.376.2.2 __num_get_type	1909
5.376.2.3 __num_put_type	1909
5.376.2.4 char_type	1910
5.376.2.5 event_callback	1910
5.376.2.6 fmtflags	1910
5.376.2.7 int_type	1911
5.376.2.8 iostate	1911
5.376.2.9 off_type	1912
5.376.2.10openmode	1912
5.376.2.11pos_type	1912
5.376.2.12seekdir	1913
5.376.2.13traits_type	1913
5.376.3 Member Enumeration Documentation	1913

5.376.3.1 event	1913
5.376.4 Constructor & Destructor Documentation	1914
5.376.4.1 basic_istream	1914
5.376.4.2 ~basic_istream	1914
5.376.5 Member Function Documentation	1914
5.376.5.1 _M_getloc	1914
5.376.5.2 bad	1914
5.376.5.3 clear	1915
5.376.5.4 copyfmt	1915
5.376.5.5 eof	1916
5.376.5.6 exceptions	1916
5.376.5.7 exceptions	1917
5.376.5.8 fail	1917
5.376.5.9 fill	1917
5.376.5.10fill	1918
5.376.5.11flags	1918
5.376.5.12flags	1918
5.376.5.13gcount	1919
5.376.5.14get	1919
5.376.5.15get	1919
5.376.5.16get	1920
5.376.5.17get	1921
5.376.5.18get	1921
5.376.5.19get	1922
5.376.5.20getline	1922
5.376.5.21getline	1923
5.376.5.22getloc	1924
5.376.5.23good	1924
5.376.5.24ignore	1924
5.376.5.25ignore	1925
5.376.5.26ignore	1925

5.376.5.27mbue	1926
5.376.5.28nit	1927
5.376.5.29word	1927
5.376.5.30narrow	1927
5.376.5.31operator void *	1928
5.376.5.32operator!	1928
5.376.5.33operator>>	1928
5.376.5.34operator>>	1929
5.376.5.35operator>>	1929
5.376.5.36operator>>	1930
5.376.5.37operator>>	1931
5.376.5.38operator>>	1931
5.376.5.39operator>>	1932
5.376.5.40operator>>	1932
5.376.5.41operator>>	1933
5.376.5.42operator>>	1934
5.376.5.43operator>>	1934
5.376.5.44operator>>	1935
5.376.5.45operator>>	1935
5.376.5.46operator>>	1936
5.376.5.47operator>>	1936
5.376.5.48operator>>	1937
5.376.5.49operator>>	1937
5.376.5.50peek	1937
5.376.5.51precision	1938
5.376.5.52precision	1938
5.376.5.53putback	1938
5.376.5.54pword	1939
5.376.5.55rdbuf	1939
5.376.5.56rdbuf	1940
5.376.5.57rdstate	1941

5.376.5.58	read	1941
5.376.5.59	readsom	1942
5.376.5.60	register_callback	1942
5.376.5.61	seekg	1943
5.376.5.62	seekg	1943
5.376.5.63	setf	1944
5.376.5.64	setf	1944
5.376.5.65	setstate	1945
5.376.5.66	sync	1945
5.376.5.67	sync_with_stdio	1946
5.376.5.68	tellg	1946
5.376.5.69	tie	1947
5.376.5.70	tie	1947
5.376.5.71	unget	1947
5.376.5.72	unsetf	1948
5.376.5.73	widen	1948
5.376.5.74	width	1949
5.376.5.75	width	1949
5.376.5.76	xalloc	1949
5.376.6	Member Data Documentation	1950
5.376.6.1	_M_gcount	1950
5.376.6.2	adjustfield	1950
5.376.6.3	app	1950
5.376.6.4	ate	1950
5.376.6.5	badbit	1950
5.376.6.6	basefield	1951
5.376.6.7	beg	1951
5.376.6.8	binary	1951
5.376.6.9	boolalpha	1951
5.376.6.10	cur	1952
5.376.6.11	dec	1952

5.376.6.12end	1952
5.376.6.13eofbit	1952
5.376.6.14failbit	1953
5.376.6.15fixed	1953
5.376.6.16floatfield	1953
5.376.6.17goodbit	1953
5.376.6.18hex	1954
5.376.6.19in	1954
5.376.6.20internal	1954
5.376.6.21left	1954
5.376.6.22oct	1954
5.376.6.23out	1955
5.376.6.24right	1955
5.376.6.25scientific	1955
5.376.6.26showbase	1955
5.376.6.27showpoint	1955
5.376.6.28showpos	1955
5.376.6.29skipws	1956
5.376.6.30trunc	1956
5.376.6.31unitbuf	1956
5.376.6.32uppercase	1956
5.377std::basic_istream<_CharT, _Traits >::sentry Class Reference	1957
5.377.1 Detailed Description	1957
5.377.2 Member Typedef Documentation	1957
5.377.2.1 traits_type	1957
5.377.3 Constructor & Destructor Documentation	1958
5.377.3.1 sentry	1958
5.377.4 Member Function Documentation	1958
5.377.4.1 operator bool	1958
5.378std::basic_istringstream<_CharT, _Traits, _Alloc > Class Template Reference	1960

5.378.1 Detailed Description	1966
5.378.2 Member Typedef Documentation	1966
5.378.2.1 __ctype_type	1966
5.378.2.2 __num_get_type	1966
5.378.2.3 __num_put_type	1967
5.378.2.4 char_type	1967
5.378.2.5 event_callback	1967
5.378.2.6 fmtflags	1967
5.378.2.7 int_type	1968
5.378.2.8 iostate	1969
5.378.2.9 off_type	1969
5.378.2.10openmode	1969
5.378.2.11pos_type	1970
5.378.2.12seekdir	1970
5.378.2.13traits_type	1970
5.378.3 Member Enumeration Documentation	1970
5.378.3.1 event	1970
5.378.4 Constructor & Destructor Documentation	1971
5.378.4.1 basic_istream	1971
5.378.4.2 basic_istream	1971
5.378.4.3 ~basic_istream	1972
5.378.5 Member Function Documentation	1972
5.378.5.1 _M_getloc	1972
5.378.5.2 bad	1972
5.378.5.3 clear	1973
5.378.5.4 copyfmt	1973
5.378.5.5 eof	1973
5.378.5.6 exceptions	1974
5.378.5.7 exceptions	1975
5.378.5.8 fail	1975
5.378.5.9 fill	1975

5.378.5.10fill	1976
5.378.5.11flags	1976
5.378.5.12flags	1976
5.378.5.13gcount	1977
5.378.5.14get	1977
5.378.5.15get	1977
5.378.5.16get	1978
5.378.5.17get	1979
5.378.5.18get	1979
5.378.5.19get	1980
5.378.5.20getline	1980
5.378.5.21getline	1981
5.378.5.22getloc	1982
5.378.5.23good	1982
5.378.5.24ignore	1982
5.378.5.25ignore	1983
5.378.5.26ignore	1983
5.378.5.27mbue	1984
5.378.5.28nit	1985
5.378.5.29word	1985
5.378.5.30narrow	1985
5.378.5.31operator void *	1986
5.378.5.32operator!	1986
5.378.5.33operator>>	1986
5.378.5.34operator>>	1987
5.378.5.35operator>>	1987
5.378.5.36operator>>	1988
5.378.5.37operator>>	1989
5.378.5.38operator>>	1989
5.378.5.39operator>>	1990
5.378.5.40operator>>	1990

5.378.5.4	operator>>	1991
5.378.5.42	operator>>	1992
5.378.5.43	operator>>	1992
5.378.5.44	operator>>	1993
5.378.5.45	operator>>	1993
5.378.5.46	operator>>	1994
5.378.5.47	operator>>	1994
5.378.5.48	operator>>	1995
5.378.5.49	operator>>	1995
5.378.5.50	peek	1995
5.378.5.51	precision	1996
5.378.5.52	precision	1996
5.378.5.53	putback	1996
5.378.5.54	pwd	1997
5.378.5.55	rdbuf	1997
5.378.5.56	rdbuf	1998
5.378.5.57	rdstate	1998
5.378.5.58	read	1999
5.378.5.59	readsome	1999
5.378.5.60	register_callback	2000
5.378.5.61	seekg	2000
5.378.5.62	seekg	2001
5.378.5.63	setf	2001
5.378.5.64	setf	2002
5.378.5.65	setstate	2002
5.378.5.66	str	2003
5.378.5.67	str	2003
5.378.5.68	sync	2003
5.378.5.69	sync_with_stdio	2004
5.378.5.70	tellg	2004
5.378.5.71	tie	2005

5.378.5.72ie	2005
5.378.5.73unget	2006
5.378.5.74unsetf	2006
5.378.5.75widen	2007
5.378.5.76width	2007
5.378.5.77width	2007
5.378.5.78xalloc	2008
5.378.6 Member Data Documentation	2008
5.378.6.1 _M_gcount	2008
5.378.6.2 adjustfield	2008
5.378.6.3 app	2009
5.378.6.4 ate	2009
5.378.6.5 badbit	2009
5.378.6.6 basefield	2009
5.378.6.7 beg	2009
5.378.6.8 binary	2010
5.378.6.9 boolalpha	2010
5.378.6.10cur	2010
5.378.6.11dec	2010
5.378.6.12end	2010
5.378.6.13eofbit	2011
5.378.6.14failbit	2011
5.378.6.15fixed	2011
5.378.6.16floatfield	2011
5.378.6.17goodbit	2012
5.378.6.18hex	2012
5.378.6.19in	2012
5.378.6.20internal	2012
5.378.6.21left	2013
5.378.6.22oct	2013
5.378.6.23out	2013

5.378.6.24	right	2013
5.378.6.25	scientific	2013
5.378.6.26	showbase	2014
5.378.6.27	showpoint	2014
5.378.6.28	showpos	2014
5.378.6.29	skipws	2014
5.378.6.30	trunc	2014
5.378.6.31	unitbuf	2014
5.378.6.32	uppercase	2014
5.379	std::basic_ofstream< _CharT, _Traits > Class Template Reference	2016
5.379.1	Detailed Description	2021
5.379.2	Member Typedef Documentation	2022
5.379.2.1	__ctype_type	2022
5.379.2.2	__num_get_type	2022
5.379.2.3	__num_put_type	2022
5.379.2.4	char_type	2022
5.379.2.5	event_callback	2023
5.379.2.6	fmtflags	2023
5.379.2.7	int_type	2024
5.379.2.8	iostate	2024
5.379.2.9	off_type	2024
5.379.2.10	openmode	2025
5.379.2.11	lpos_type	2025
5.379.2.12	seekdir	2025
5.379.2.13	traits_type	2026
5.379.3	Member Enumeration Documentation	2026
5.379.3.1	event	2026
5.379.4	Constructor & Destructor Documentation	2026
5.379.4.1	basic_ofstream	2026
5.379.4.2	basic_ofstream	2026
5.379.4.3	basic_ofstream	2027

5.379.4.4 ~basic_ofstream	2027
5.379.5 Member Function Documentation	2027
5.379.5.1 _M_getloc	2027
5.379.5.2 _M_write	2028
5.379.5.3 bad	2028
5.379.5.4 clear	2028
5.379.5.5 close	2029
5.379.5.6 copyfmt	2029
5.379.5.7 eof	2029
5.379.5.8 exceptions	2030
5.379.5.9 exceptions	2030
5.379.5.10fail	2031
5.379.5.11fill	2031
5.379.5.12fill	2032
5.379.5.13flags	2032
5.379.5.14flags	2032
5.379.5.15flush	2033
5.379.5.16getloc	2033
5.379.5.17good	2033
5.379.5.18mbue	2034
5.379.5.19nit	2034
5.379.5.20s_open	2034
5.379.5.21word	2035
5.379.5.22narrow	2035
5.379.5.23open	2036
5.379.5.24open	2036
5.379.5.25operator void *	2036
5.379.5.26operator!	2037
5.379.5.27operator<<	2037
5.379.5.28operator<<	2037
5.379.5.29operator<<	2038

5.379.5.30operator<<	2039
5.379.5.31operator<<	2039
5.379.5.32operator<<	2040
5.379.5.33operator<<	2041
5.379.5.34operator<<	2041
5.379.5.35operator<<	2042
5.379.5.36operator<<	2042
5.379.5.37operator<<	2043
5.379.5.38operator<<	2044
5.379.5.39operator<<	2044
5.379.5.40operator<<	2045
5.379.5.41operator<<	2045
5.379.5.42operator<<	2046
5.379.5.43operator<<	2046
5.379.5.44precision	2046
5.379.5.45precision	2046
5.379.5.46put	2047
5.379.5.47pword	2047
5.379.5.48&dbuf	2048
5.379.5.49&dbuf	2048
5.379.5.50rdstate	2049
5.379.5.51register_callback	2049
5.379.5.52seekp	2049
5.379.5.53seekp	2050
5.379.5.54setf	2050
5.379.5.55setf	2051
5.379.5.56setstate	2051
5.379.5.57sync_with_stdio	2052
5.379.5.58tellp	2052
5.379.5.59tie	2052
5.379.5.60tie	2053

5.379.5.61unsetf	2053
5.379.5.62widen	2053
5.379.5.63width	2054
5.379.5.64width	2054
5.379.5.65write	2055
5.379.5.66xalloc	2055
5.379.6 Member Data Documentation	2056
5.379.6.1 adjustfield	2056
5.379.6.2 app	2056
5.379.6.3 ate	2056
5.379.6.4 badbit	2056
5.379.6.5 basefield	2057
5.379.6.6 beg	2057
5.379.6.7 binary	2057
5.379.6.8 boolalpha	2057
5.379.6.9 cur	2057
5.379.6.10dec	2058
5.379.6.11lend	2058
5.379.6.12eofbit	2058
5.379.6.13failbit	2058
5.379.6.14fixed	2059
5.379.6.15floatfield	2059
5.379.6.16goodbit	2059
5.379.6.17hex	2059
5.379.6.18n	2059
5.379.6.19internal	2060
5.379.6.20left	2060
5.379.6.21loct	2060
5.379.6.22out	2060
5.379.6.23right	2061
5.379.6.24scientific	2061

5.379.6.25	showbase	2061
5.379.6.26	showpoint	2061
5.379.6.27	showpos	2061
5.379.6.28	skipws	2061
5.379.6.29	trunc	2061
5.379.6.30	unitbuf	2062
5.379.6.3	uppercase	2062
5.380	std::basic_ostream< _CharT, _Traits > Class Template Reference	2063
5.380.1	Detailed Description	2068
5.380.2	Member Typedef Documentation	2068
5.380.2.1	__ctype_type	2068
5.380.2.2	__num_get_type	2068
5.380.2.3	__num_put_type	2069
5.380.2.4	char_type	2069
5.380.2.5	event_callback	2069
5.380.2.6	fmtflags	2069
5.380.2.7	int_type	2070
5.380.2.8	iostate	2071
5.380.2.9	off_type	2071
5.380.2.10	openmode	2071
5.380.2.11	lpos_type	2072
5.380.2.12	seekdir	2072
5.380.2.13	traits_type	2072
5.380.3	Member Enumeration Documentation	2072
5.380.3.1	event	2072
5.380.4	Constructor & Destructor Documentation	2073
5.380.4.1	basic_ostream	2073
5.380.4.2	~basic_ostream	2073
5.380.5	Member Function Documentation	2073
5.380.5.1	_M_getloc	2073
5.380.5.2	_M_write	2074

5.380.5.3 bad	2074
5.380.5.4 clear	2074
5.380.5.5 copyfmt	2075
5.380.5.6 eof	2075
5.380.5.7 exceptions	2076
5.380.5.8 exceptions	2076
5.380.5.9 fail	2077
5.380.5.10fill	2077
5.380.5.11fill	2077
5.380.5.12flags	2078
5.380.5.13flags	2078
5.380.5.14flush	2078
5.380.5.15getloc	2079
5.380.5.16good	2079
5.380.5.17mbue	2079
5.380.5.18nit	2080
5.380.5.19word	2080
5.380.5.20narrow	2081
5.380.5.21operator void *	2081
5.380.5.22operator!	2081
5.380.5.23operator<<	2082
5.380.5.24operator<<	2082
5.380.5.25operator<<	2083
5.380.5.26operator<<	2083
5.380.5.27operator<<	2084
5.380.5.28operator<<	2085
5.380.5.29operator<<	2085
5.380.5.30operator<<	2086
5.380.5.31operator<<	2087
5.380.5.32operator<<	2087
5.380.5.33operator<<	2088

5.380.5.34operator<<	2088
5.380.5.35operator<<	2089
5.380.5.36operator<<	2090
5.380.5.37operator<<	2090
5.380.5.38operator<<	2090
5.380.5.39operator<<	2091
5.380.5.40precision	2091
5.380.5.41precision	2091
5.380.5.42put	2091
5.380.5.43pword	2092
5.380.5.44rdbuf	2092
5.380.5.45rdbuf	2093
5.380.5.46rdstate	2094
5.380.5.47register_callback	2094
5.380.5.48seekp	2094
5.380.5.49seekp	2095
5.380.5.50setf	2095
5.380.5.51setf	2096
5.380.5.52setstate	2096
5.380.5.53sync_with_stdio	2097
5.380.5.54tellp	2097
5.380.5.55tie	2097
5.380.5.56tie	2098
5.380.5.57unsetf	2098
5.380.5.58widen	2099
5.380.5.59width	2099
5.380.5.60width	2099
5.380.5.61write	2100
5.380.5.62xalloc	2100
5.380.6 Member Data Documentation	2101
5.380.6.1 adjustfield	2101

5.380.6.2	app	2101
5.380.6.3	ate	2101
5.380.6.4	badbit	2101
5.380.6.5	basefield	2102
5.380.6.6	beg	2102
5.380.6.7	binary	2102
5.380.6.8	boolalpha	2102
5.380.6.9	cur	2102
5.380.6.10	dec	2103
5.380.6.11	end	2103
5.380.6.12	eofbit	2103
5.380.6.13	failbit	2103
5.380.6.14	fixed	2104
5.380.6.15	floatfield	2104
5.380.6.16	goodbit	2104
5.380.6.17	hex	2104
5.380.6.18	in	2105
5.380.6.19	internal	2105
5.380.6.20	left	2105
5.380.6.21	loct	2105
5.380.6.22	out	2105
5.380.6.23	right	2106
5.380.6.24	scientific	2106
5.380.6.25	showbase	2106
5.380.6.26	showpoint	2106
5.380.6.27	showpos	2106
5.380.6.28	skipws	2106
5.380.6.29	trunc	2106
5.380.6.30	unitbuf	2107
5.380.6.31	uppercase	2107
5.381	std::basic_ostream<_CharT, _Traits >::sentry Class Reference	2108

5.381.1 Detailed Description	2108
5.381.2 Constructor & Destructor Documentation	2108
5.381.2.1 sentry	2108
5.381.2.2 ~sentry	2109
5.381.3 Member Function Documentation	2109
5.381.3.1 operator bool	2109
5.382std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	2110
5.382.1 Detailed Description	2115
5.382.2 Member Typedef Documentation	2116
5.382.2.1 __ctype_type	2116
5.382.2.2 __num_get_type	2116
5.382.2.3 __num_put_type	2116
5.382.2.4 char_type	2116
5.382.2.5 event_callback	2117
5.382.2.6 fmtflags	2117
5.382.2.7 int_type	2118
5.382.2.8 iostate	2118
5.382.2.9 off_type	2118
5.382.2.10openmode	2119
5.382.2.11pos_type	2119
5.382.2.12seekdir	2119
5.382.2.13traits_type	2120
5.382.3 Member Enumeration Documentation	2120
5.382.3.1 event	2120
5.382.4 Constructor & Destructor Documentation	2120
5.382.4.1 basic_ostringstream	2120
5.382.4.2 basic_ostringstream	2121
5.382.4.3 ~basic_ostringstream	2121
5.382.5 Member Function Documentation	2121
5.382.5.1 _M_getloc	2121

5.382.5.2	_M_write	2122
5.382.5.3	bad	2122
5.382.5.4	clear	2122
5.382.5.5	copyfmt	2123
5.382.5.6	eof	2123
5.382.5.7	exceptions	2124
5.382.5.8	exceptions	2124
5.382.5.9	fail	2125
5.382.5.10	fill	2125
5.382.5.11	fill	2126
5.382.5.12	flags	2126
5.382.5.13	flags	2126
5.382.5.14	flush	2127
5.382.5.15	getloc	2127
5.382.5.16	good	2127
5.382.5.17	mbue	2128
5.382.5.18	nit	2128
5.382.5.19	word	2128
5.382.5.20	narrow	2129
5.382.5.21	operator void *	2129
5.382.5.22	operator!	2130
5.382.5.23	operator<<	2130
5.382.5.24	operator<<	2130
5.382.5.25	operator<<	2131
5.382.5.26	operator<<	2132
5.382.5.27	operator<<	2132
5.382.5.28	operator<<	2133
5.382.5.29	operator<<	2134
5.382.5.30	operator<<	2134
5.382.5.31	operator<<	2135
5.382.5.32	operator<<	2135

5.382.5.33operator<<	2136
5.382.5.34operator<<	2137
5.382.5.35operator<<	2137
5.382.5.36operator<<	2138
5.382.5.37operator<<	2138
5.382.5.38operator<<	2139
5.382.5.39operator<<	2139
5.382.5.40precision	2139
5.382.5.41precision	2139
5.382.5.42put	2140
5.382.5.43pword	2140
5.382.5.44rdbuf	2141
5.382.5.45rdbuf	2141
5.382.5.46rdstate	2142
5.382.5.47register_callback	2142
5.382.5.48seekp	2142
5.382.5.49seekp	2143
5.382.5.50setf	2143
5.382.5.51setf	2144
5.382.5.52setstate	2144
5.382.5.53str	2145
5.382.5.54str	2145
5.382.5.55sync_with_stdio	2145
5.382.5.56tellp	2146
5.382.5.57tie	2146
5.382.5.58tie	2146
5.382.5.59unsetf	2147
5.382.5.60widen	2147
5.382.5.61width	2148
5.382.5.62width	2148
5.382.5.63write	2148

5.382.5.64xalloc	2149
5.382.6 Member Data Documentation	2149
5.382.6.1 adjustfield	2149
5.382.6.2 app	2149
5.382.6.3 ate	2150
5.382.6.4 badbit	2150
5.382.6.5 basefield	2150
5.382.6.6 beg	2150
5.382.6.7 binary	2151
5.382.6.8 boolalpha	2151
5.382.6.9 cur	2151
5.382.6.10dec	2151
5.382.6.11end	2151
5.382.6.12eofbit	2152
5.382.6.13failbit	2152
5.382.6.14fixed	2152
5.382.6.15floatfield	2152
5.382.6.16goodbit	2153
5.382.6.17hex	2153
5.382.6.18n	2153
5.382.6.19internal	2153
5.382.6.20left	2154
5.382.6.21loct	2154
5.382.6.22out	2154
5.382.6.23right	2154
5.382.6.24scientific	2154
5.382.6.25showbase	2155
5.382.6.26showpoint	2155
5.382.6.27showpos	2155
5.382.6.28skipws	2155
5.382.6.29trunc	2155

5.382.6.30unitbuf	2155
5.382.6.3 luppercase	2155
5.383std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	2157
5.383.1 Detailed Description	2158
5.383.2 Constructor & Destructor Documentation	2159
5.383.2.1 basic_regex	2159
5.383.2.2 basic_regex	2159
5.383.2.3 basic_regex	2159
5.383.2.4 basic_regex	2160
5.383.2.5 basic_regex	2160
5.383.2.6 basic_regex	2161
5.383.2.7 basic_regex	2161
5.383.2.8 ~basic_regex	2161
5.383.3 Member Function Documentation	2162
5.383.3.1 assign	2162
5.383.3.2 assign	2162
5.383.3.3 assign	2163
5.383.3.4 assign	2163
5.383.3.5 assign	2164
5.383.3.6 assign	2164
5.383.3.7 flags	2164
5.383.3.8 getloc	2165
5.383.3.9 imbue	2165
5.383.3.10mark_count	2165
5.383.3.11operator=	2165
5.383.3.12operator=	2166
5.383.3.13operator=	2166
5.383.3.14swap	2166
5.384std::basic_streambuf< _CharT, _Traits > Class Template Reference	2168
5.384.1 Detailed Description	2170
5.384.2 Member Typedef Documentation	2172

5.384.2.1	<code>__streambuf_type</code>	2172
5.384.2.2	<code>char_type</code>	2172
5.384.2.3	<code>int_type</code>	2172
5.384.2.4	<code>off_type</code>	2172
5.384.2.5	<code>pos_type</code>	2173
5.384.2.6	<code>traits_type</code>	2173
5.384.3	Constructor & Destructor Documentation	2173
5.384.3.1	<code>~basic_streambuf</code>	2173
5.384.3.2	<code>basic_streambuf</code>	2174
5.384.4	Member Function Documentation	2174
5.384.4.1	<code>eback</code>	2174
5.384.4.2	<code>egptr</code>	2174
5.384.4.3	<code>epptr</code>	2175
5.384.4.4	<code>gbump</code>	2175
5.384.4.5	<code>getloc</code>	2176
5.384.4.6	<code>gptr</code>	2176
5.384.4.7	<code>imbue</code>	2176
5.384.4.8	<code>in_avail</code>	2177
5.384.4.9	<code>overflow</code>	2177
5.384.4.10	<code>backfail</code>	2178
5.384.4.11	<code>lbase</code>	2179
5.384.4.12	<code>pbump</code>	2179
5.384.4.13	<code>pptr</code>	2179
5.384.4.14	<code>pubimbue</code>	2180
5.384.4.15	<code>pubseekoff</code>	2180
5.384.4.16	<code>pubseekpos</code>	2181
5.384.4.17	<code>pubsetbuf</code>	2181
5.384.4.18	<code>pubsync</code>	2181
5.384.4.19	<code>sbumpc</code>	2181
5.384.4.20	<code>seekoff</code>	2182
5.384.4.21	<code>seekpos</code>	2182

5.384.4.22	setbuf	2183
5.384.4.23	setg	2183
5.384.4.24	setp	2183
5.384.4.25	getc	2184
5.384.4.26	getn	2184
5.384.4.27	showmanyc	2185
5.384.4.28	nextc	2185
5.384.4.29	putbackc	2186
5.384.4.30	putc	2186
5.384.4.31	lputn	2187
5.384.4.32	stoss	2187
5.384.4.33	ungetc	2187
5.384.4.34	sync	2188
5.384.4.35	uflow	2188
5.384.4.36	underflow	2188
5.384.4.37	xsgetn	2189
5.384.4.38	xspn	2190
5.384.5	Member Data Documentation	2190
5.384.5.1	_M_buf_locale	2190
5.384.5.2	_M_in_beg	2191
5.384.5.3	_M_in_cur	2191
5.384.5.4	_M_in_end	2191
5.384.5.5	_M_out_beg	2192
5.384.5.6	_M_out_cur	2192
5.384.5.7	_M_out_end	2192
5.385	std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference	2193
5.385.1	Detailed Description	2197
5.385.2	Constructor & Destructor Documentation	2198
5.385.2.1	basic_string	2198
5.385.2.2	basic_string	2198
5.385.2.3	basic_string	2199

5.385.2.4	basic_string	2199
5.385.2.5	basic_string	2199
5.385.2.6	basic_string	2200
5.385.2.7	basic_string	2200
5.385.2.8	basic_string	2200
5.385.2.9	basic_string	2201
5.385.2.10	basic_string	2201
5.385.2.11	basic_string	2201
5.385.2.12	~basic_string	2202
5.385.3	Member Function Documentation	2202
5.385.3.1	append	2202
5.385.3.2	append	2202
5.385.3.3	append	2203
5.385.3.4	append	2203
5.385.3.5	append	2204
5.385.3.6	append	2204
5.385.3.7	append	2205
5.385.3.8	assign	2205
5.385.3.9	assign	2206
5.385.3.10	assign	2206
5.385.3.11	lassign	2206
5.385.3.12	assign	2207
5.385.3.13	assign	2207
5.385.3.14	assign	2208
5.385.3.15	assign	2208
5.385.3.16	at	2209
5.385.3.17	at	2209
5.385.3.18	begin	2210
5.385.3.19	begin	2210
5.385.3.20	c_str	2210
5.385.3.21	capacity	2211

5.385.3.22 begin	2211
5.385.3.23 end	2211
5.385.3.24 clear	2211
5.385.3.25 compare	2212
5.385.3.26 compare	2212
5.385.3.27 compare	2213
5.385.3.28 compare	2213
5.385.3.29 compare	2214
5.385.3.30 compare	2215
5.385.3.3 lcopy	2215
5.385.3.32 rbegin	2216
5.385.3.33 rend	2216
5.385.3.34 data	2216
5.385.3.35 empty	2217
5.385.3.36 end	2217
5.385.3.37 end	2217
5.385.3.38 erase	2217
5.385.3.39 erase	2218
5.385.3.40 erase	2218
5.385.3.41 find	2219
5.385.3.42 find	2219
5.385.3.43 find	2220
5.385.3.44 find	2220
5.385.3.45 find_first_not_of	2221
5.385.3.46 find_first_not_of	2221
5.385.3.47 find_first_not_of	2222
5.385.3.48 find_first_not_of	2222
5.385.3.49 find_first_of	2223
5.385.3.50 find_first_of	2223
5.385.3.51 find_first_of	2224
5.385.3.52 find_first_of	2224

5.385.3.53	find_last_not_of	2225
5.385.3.54	find_last_not_of	2225
5.385.3.55	find_last_not_of	2226
5.385.3.56	find_last_not_of	2226
5.385.3.57	find_last_of	2227
5.385.3.58	find_last_of	2227
5.385.3.59	find_last_of	2227
5.385.3.60	find_last_of	2228
5.385.3.61	get_allocator	2228
5.385.3.62	insert	2229
5.385.3.63	insert	2229
5.385.3.64	insert	2230
5.385.3.65	insert	2230
5.385.3.66	insert	2231
5.385.3.67	insert	2232
5.385.3.68	insert	2232
5.385.3.69	insert	2233
5.385.3.70	insert	2233
5.385.3.71	length	2234
5.385.3.72	max_size	2234
5.385.3.73	operator+=	2234
5.385.3.74	operator+=	2234
5.385.3.75	operator+=	2235
5.385.3.76	operator+=	2235
5.385.3.77	operator=	2236
5.385.3.78	operator=	2236
5.385.3.79	operator=	2236
5.385.3.80	operator=	2237
5.385.3.81	operator=	2237
5.385.3.82	operator[]	2237
5.385.3.83	operator[]	2238

5.385.3.84	push_back	2238
5.385.3.85	begin	2238
5.385.3.86	begin	2239
5.385.3.87	end	2239
5.385.3.88	end	2239
5.385.3.89	replace	2239
5.385.3.90	replace	2240
5.385.3.91	replace	2241
5.385.3.92	replace	2241
5.385.3.93	replace	2242
5.385.3.94	replace	2242
5.385.3.95	replace	2243
5.385.3.96	replace	2244
5.385.3.97	replace	2244
5.385.3.98	replace	2245
5.385.3.99	replace	2246
5.385.3.100	reserve	2246
5.385.3.101	size	2247
5.385.3.102	size	2247
5.385.3.103	find	2248
5.385.3.104	find	2248
5.385.3.105	find	2249
5.385.3.106	find	2249
5.385.3.107	shrink_to_fit	2250
5.385.3.108	size	2250
5.385.3.109	substr	2250
5.385.3.110	swap	2251
5.385.4	Member Data Documentation	2251
5.385.4.1	npos	2251
5.386	std::basic_stringbuf<_CharT, _Traits, _Alloc > Class Template Reference	2252

5.386.1 Detailed Description	2255
5.386.2 Member Typedef Documentation	2255
5.386.2.1 __streambuf_type	2255
5.386.2.2 char_type	2255
5.386.2.3 int_type	2256
5.386.2.4 off_type	2256
5.386.2.5 pos_type	2256
5.386.2.6 traits_type	2256
5.386.3 Constructor & Destructor Documentation	2257
5.386.3.1 basic_stringbuf	2257
5.386.3.2 basic_stringbuf	2257
5.386.4 Member Function Documentation	2257
5.386.4.1 eback	2257
5.386.4.2 egptr	2258
5.386.4.3 epptr	2258
5.386.4.4 gbump	2259
5.386.4.5 getloc	2259
5.386.4.6 gptr	2259
5.386.4.7 imbue	2260
5.386.4.8 in_avail	2260
5.386.4.9 overflow	2261
5.386.4.10pbackfail	2262
5.386.4.11pbase	2262
5.386.4.12pbump	2263
5.386.4.13pptr	2263
5.386.4.14pubimbue	2263
5.386.4.15pubseekoff	2264
5.386.4.16pubseekpos	2264
5.386.4.17pubsetbuf	2264
5.386.4.18pubsync	2265
5.386.4.19sbumpc	2265

5.386.4.20	seekoff	2265
5.386.4.21	seekpos	2266
5.386.4.22	setbuf	2266
5.386.4.23	setg	2267
5.386.4.24	setp	2267
5.386.4.25	setc	2268
5.386.4.26	setn	2268
5.386.4.27	showmanyc	2268
5.386.4.28	nextc	2269
5.386.4.29	putbackc	2269
5.386.4.30	putc	2270
5.386.4.31	lputn	2270
5.386.4.32	stossc	2271
5.386.4.33	str	2271
5.386.4.34	str	2271
5.386.4.35	sungetc	2272
5.386.4.36	sync	2272
5.386.4.37	uflow	2272
5.386.4.38	underflow	2273
5.386.4.39	xsetn	2274
5.386.4.40	xputn	2274
5.386.5	Member Data Documentation	2275
5.386.5.1	_M_buf_locale	2275
5.386.5.2	_M_in_beg	2275
5.386.5.3	_M_in_cur	2275
5.386.5.4	_M_in_end	2276
5.386.5.5	_M_mode	2276
5.386.5.6	_M_out_beg	2276
5.386.5.7	_M_out_cur	2277
5.386.5.8	_M_out_end	2277

5.387std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template	
Reference	2278
5.387.1 Detailed Description	2285
5.387.2 Member Typedef Documentation	2286
5.387.2.1 __ctype_type	2286
5.387.2.2 __ctype_type	2286
5.387.2.3 __num_get_type	2286
5.387.2.4 __num_put_type	2286
5.387.2.5 __num_put_type	2287
5.387.2.6 char_type	2287
5.387.2.7 event_callback	2287
5.387.2.8 fmtflags	2287
5.387.2.9 int_type	2288
5.387.2.10iostate	2289
5.387.2.11loff_type	2289
5.387.2.12openmode	2289
5.387.2.13pos_type	2290
5.387.2.14seekdir	2290
5.387.2.15traits_type	2290
5.387.3 Member Enumeration Documentation	2290
5.387.3.1 event	2290
5.387.4 Constructor & Destructor Documentation	2291
5.387.4.1 basic_stringstream	2291
5.387.4.2 basic_stringstream	2291
5.387.4.3 ~basic_stringstream	2291
5.387.5 Member Function Documentation	2292
5.387.5.1 _M_getloc	2292
5.387.5.2 _M_write	2292
5.387.5.3 bad	2292
5.387.5.4 clear	2293
5.387.5.5 copyfmt	2293

5.387.5.6 eof	2294
5.387.5.7 exceptions	2294
5.387.5.8 exceptions	2295
5.387.5.9 fail	2295
5.387.5.10fill	2295
5.387.5.11fill	2296
5.387.5.12flags	2296
5.387.5.13flags	2296
5.387.5.14flush	2297
5.387.5.15gcount	2297
5.387.5.16get	2297
5.387.5.17get	2298
5.387.5.18get	2299
5.387.5.19get	2299
5.387.5.20get	2300
5.387.5.21get	2300
5.387.5.22getline	2301
5.387.5.23getline	2301
5.387.5.24getloc	2302
5.387.5.25good	2302
5.387.5.26ignore	2303
5.387.5.27ignore	2303
5.387.5.28ignore	2304
5.387.5.29mbue	2304
5.387.5.30nit	2305
5.387.5.31word	2305
5.387.5.32narrow	2306
5.387.5.33operator void *	2306
5.387.5.34operator!	2306
5.387.5.35operator<<	2307
5.387.5.36operator<<	2307

5.387.5.37operator<<	2308
5.387.5.38operator<<	2309
5.387.5.39operator<<	2309
5.387.5.40operator<<	2310
5.387.5.41operator<<	2310
5.387.5.42operator<<	2311
5.387.5.43operator<<	2312
5.387.5.44operator<<	2312
5.387.5.45operator<<	2313
5.387.5.46operator<<	2313
5.387.5.47operator<<	2314
5.387.5.48operator<<	2315
5.387.5.49operator<<	2315
5.387.5.50operator<<	2315
5.387.5.51operator<<	2316
5.387.5.52operator>>	2316
5.387.5.53operator>>	2317
5.387.5.54operator>>	2317
5.387.5.55operator>>	2318
5.387.5.56operator>>	2318
5.387.5.57operator>>	2319
5.387.5.58operator>>	2320
5.387.5.59operator>>	2320
5.387.5.60operator>>	2321
5.387.5.61operator>>	2321
5.387.5.62operator>>	2322
5.387.5.63operator>>	2323
5.387.5.64operator>>	2323
5.387.5.65operator>>	2324
5.387.5.66operator>>	2324
5.387.5.67operator>>	2324

5.387.5.68operator>>	2325
5.387.5.69peek	2325
5.387.5.70precision	2325
5.387.5.71precision	2326
5.387.5.72put	2326
5.387.5.73putback	2326
5.387.5.74pword	2327
5.387.5.75rdbuf	2328
5.387.5.76rdbuf	2328
5.387.5.77rdstate	2329
5.387.5.78read	2329
5.387.5.79readsome	2330
5.387.5.80register_callback	2330
5.387.5.81seekg	2331
5.387.5.82seekg	2331
5.387.5.83seekp	2332
5.387.5.84seekp	2332
5.387.5.85setf	2333
5.387.5.86setf	2333
5.387.5.87setstate	2334
5.387.5.88str	2334
5.387.5.89str	2334
5.387.5.90sync	2335
5.387.5.91sync_with_stdio	2335
5.387.5.92tellg	2336
5.387.5.93tellp	2336
5.387.5.94tie	2336
5.387.5.95tie	2337
5.387.5.96unget	2337
5.387.5.97unsetf	2338
5.387.5.98widen	2338

5.387.5.9width	2339
5.387.5.10width	2339
5.387.5.10write	2339
5.387.5.10zalloc	2340
5.387.6 Member Data Documentation	2340
5.387.6.1 _M_gcount	2340
5.387.6.2 adjustfield	2341
5.387.6.3 app	2341
5.387.6.4 ate	2341
5.387.6.5 badbit	2341
5.387.6.6 basefield	2342
5.387.6.7 beg	2342
5.387.6.8 binary	2342
5.387.6.9 boolalpha	2342
5.387.6.10cur	2342
5.387.6.11dec	2343
5.387.6.12end	2343
5.387.6.13eofbit	2343
5.387.6.14failbit	2343
5.387.6.15fixed	2344
5.387.6.16floatfield	2344
5.387.6.17goodbit	2344
5.387.6.18hex	2344
5.387.6.19n	2344
5.387.6.20internal	2345
5.387.6.21left	2345
5.387.6.22oct	2345
5.387.6.23out	2345
5.387.6.24right	2346
5.387.6.25scientific	2346
5.387.6.26showbase	2346

5.387.6.27	showpoint	2346
5.387.6.28	showpos	2346
5.387.6.29	skipws	2346
5.387.6.30	trunc	2346
5.387.6.31	unitbuf	2347
5.387.6.32	uppercase	2347
5.388	std::bernoulli_distribution Class Reference	2348
5.388.1	Detailed Description	2348
5.388.2	Member Typedef Documentation	2348
5.388.2.1	result_type	2348
5.388.3	Constructor & Destructor Documentation	2349
5.388.3.1	bernoulli_distribution	2349
5.388.4	Member Function Documentation	2349
5.388.4.1	max	2349
5.388.4.2	min	2349
5.388.4.3	operator()	2349
5.388.4.4	p	2349
5.388.4.5	param	2350
5.388.4.6	param	2350
5.388.4.7	reset	2350
5.389	std::bernoulli_distribution::param_type Struct Reference	2351
5.389.1	Detailed Description	2351
5.390	std::bidirectional_iterator_tag Struct Reference	2352
5.390.1	Detailed Description	2352
5.391	std::binary_function<_Arg1, _Arg2, _Result> Struct Template Reference	2353
5.391.1	Detailed Description	2353
5.391.2	Member Typedef Documentation	2354
5.391.2.1	first_argument_type	2354
5.391.2.2	result_type	2354
5.391.2.3	second_argument_type	2354

5.392std::binary_negate< _Predicate > Class Template Reference	2355
5.392.1 Detailed Description	2355
5.392.2 Member Typedef Documentation	2355
5.392.2.1 first_argument_type	2355
5.392.2.2 result_type	2356
5.392.2.3 second_argument_type	2356
5.393std::binder1st< _Operation > Class Template Reference	2357
5.393.1 Detailed Description	2357
5.393.2 Member Typedef Documentation	2358
5.393.2.1 argument_type	2358
5.393.2.2 result_type	2358
5.394std::binder2nd< _Operation > Class Template Reference	2359
5.394.1 Detailed Description	2359
5.394.2 Member Typedef Documentation	2360
5.394.2.1 argument_type	2360
5.394.2.2 result_type	2360
5.395std::binomial_distribution< _IntType > Class Template Reference . .	2361
5.395.1 Detailed Description	2362
5.395.2 Member Typedef Documentation	2362
5.395.2.1 result_type	2362
5.395.3 Member Function Documentation	2362
5.395.3.1 max	2362
5.395.3.2 min	2362
5.395.3.3 operator()	2363
5.395.3.4 p	2363
5.395.3.5 param	2363
5.395.3.6 param	2363
5.395.3.7 reset	2363
5.395.3.8 t	2364
5.395.4 Friends And Related Function Documentation	2364
5.395.4.1 operator<<	2364

5.395.4.2 operator>>	2364
5.396std::binomial_distribution<_IntType>::param_type Struct Reference	2366
5.396.1 Detailed Description	2366
5.397std::bitset<_Nb> Class Template Reference	2367
5.397.1 Detailed Description	2370
5.397.2 Constructor & Destructor Documentation	2371
5.397.2.1 bitset	2371
5.397.2.2 bitset	2371
5.397.2.3 bitset	2371
5.397.2.4 bitset	2372
5.397.2.5 bitset	2372
5.397.3 Member Function Documentation	2373
5.397.3.1 _Unchecked_flip	2373
5.397.3.2 _Unchecked_reset	2373
5.397.3.3 _Unchecked_set	2373
5.397.3.4 _Unchecked_test	2373
5.397.3.5 all	2373
5.397.3.6 any	2374
5.397.3.7 count	2374
5.397.3.8 flip	2374
5.397.3.9 flip	2374
5.397.3.10none	2374
5.397.3.11operator!=	2375
5.397.3.12operator&=	2375
5.397.3.13operator<<	2375
5.397.3.14operator<<=	2375
5.397.3.15operator==	2376
5.397.3.16operator>>	2376
5.397.3.17operator>>=	2376
5.397.3.18operator[]	2376
5.397.3.19operator[]	2376

5.397.3.20operator [^] =	2377
5.397.3.21operator =	2377
5.397.3.22operator~	2377
5.397.3.23reset	2377
5.397.3.24reset	2378
5.397.3.25set	2378
5.397.3.26set	2378
5.397.3.27size	2378
5.397.3.28test	2378
5.397.3.29to_string	2379
5.397.3.30to_ulong	2379
5.398std::bitset<_Nb>::reference Class Reference	2380
5.398.1 Detailed Description	2380
5.399std::cauchy_distribution<_RealType> Class Template Reference	2381
5.399.1 Detailed Description	2381
5.399.2 Member Typedef Documentation	2382
5.399.2.1 result_type	2382
5.399.3 Member Function Documentation	2382
5.399.3.1 max	2382
5.399.3.2 min	2382
5.399.3.3 param	2382
5.399.3.4 param	2382
5.399.3.5 reset	2383
5.400std::cauchy_distribution<_RealType>::param_type Struct Reference	2384
5.400.1 Detailed Description	2384
5.401std::char_traits<_CharT> Struct Template Reference	2385
5.401.1 Detailed Description	2386
5.402std::char_traits<__gnu_cxx::character<V, I, S>> Struct Template Reference	2387
5.402.1 Detailed Description	2387
5.403std::char_traits<char> Struct Template Reference	2388

5.403.1 Detailed Description	2388
5.404std::char_traits< wchar_t > Struct Template Reference	2389
5.404.1 Detailed Description	2389
5.405std::chi_squared_distribution< _RealType > Class Template Reference	2390
5.405.1 Detailed Description	2391
5.405.2 Member Typedef Documentation	2391
5.405.2.1 result_type	2391
5.405.3 Member Function Documentation	2391
5.405.3.1 max	2391
5.405.3.2 min	2391
5.405.3.3 param	2391
5.405.3.4 param	2392
5.405.3.5 reset	2392
5.405.4 Friends And Related Function Documentation	2392
5.405.4.1 operator<<	2392
5.405.4.2 operator>>	2393
5.406std::chi_squared_distribution< _RealType >::param_type Struct Ref-	
erence	2394
5.406.1 Detailed Description	2394
5.407std::chrono::duration< _Rep, _Period > Struct Template Reference .	2395
5.407.1 Detailed Description	2396
5.408std::chrono::duration_values< _Rep > Struct Template Reference . .	2397
5.408.1 Detailed Description	2397
5.409std::chrono::system_clock Struct Reference	2398
5.409.1 Detailed Description	2398
5.410std::chrono::time_point< _Clock, _Duration > Struct Template Refer-	
erence	2399
5.410.1 Detailed Description	2399
5.411std::chrono::treat_as_floating_point< _Rep > Struct Template Reference	2400
5.411.1 Detailed Description	2400
5.412std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	2401
5.412.1 Detailed Description	2403

5.412.2 Member Function Documentation	2403
5.412.2.1 do_out	2403
5.412.2.2 in	2404
5.412.2.3 out	2405
5.412.2.4 unshift	2406
5.413std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	2407
5.413.1 Detailed Description	2408
5.413.2 Member Function Documentation	2409
5.413.2.1 do_out	2409
5.413.2.2 in	2409
5.413.2.3 out	2410
5.413.2.4 unshift	2411
5.414std::codecvt< char, char, mbstate_t > Class Template Reference . . .	2413
5.414.1 Detailed Description	2414
5.414.2 Member Function Documentation	2415
5.414.2.1 do_out	2415
5.414.2.2 in	2415
5.414.2.3 out	2416
5.414.2.4 unshift	2417
5.415std::codecvt< wchar_t, char, mbstate_t > Class Template Reference .	2419
5.415.1 Detailed Description	2420
5.415.2 Member Function Documentation	2421
5.415.2.1 do_out	2421
5.415.2.2 in	2421
5.415.2.3 out	2422
5.415.2.4 unshift	2423
5.416std::codecvt_base Class Reference	2425
5.416.1 Detailed Description	2425
5.417std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	2426
5.417.1 Detailed Description	2428

5.417.2 Member Function Documentation	2428
5.417.2.1 do_out	2428
5.417.2.2 in	2428
5.417.2.3 out	2429
5.417.2.4 unshift	2430
5.418std::collate< _CharT > Class Template Reference	2432
5.418.1 Detailed Description	2433
5.418.2 Member Typedef Documentation	2434
5.418.2.1 char_type	2434
5.418.2.2 string_type	2434
5.418.3 Constructor & Destructor Documentation	2434
5.418.3.1 collate	2434
5.418.3.2 collate	2434
5.418.3.3 ~collate	2435
5.418.4 Member Function Documentation	2435
5.418.4.1 compare	2435
5.418.4.2 do_compare	2435
5.418.4.3 do_hash	2436
5.418.4.4 do_transform	2436
5.418.4.5 hash	2437
5.418.4.6 transform	2437
5.418.5 Member Data Documentation	2438
5.418.5.1 id	2438
5.419std::collate_byname< _CharT > Class Template Reference	2439
5.419.1 Detailed Description	2440
5.419.2 Member Typedef Documentation	2441
5.419.2.1 char_type	2441
5.419.2.2 string_type	2441
5.419.3 Member Function Documentation	2441
5.419.3.1 compare	2441
5.419.3.2 do_compare	2442

5.419.3.3 do_hash	2442
5.419.3.4 do_transform	2443
5.419.3.5 hash	2443
5.419.3.6 transform	2444
5.419.4 Member Data Documentation	2444
5.419.4.1 id	2444
5.420std::complex<_Tp> Struct Template Reference	2445
5.420.1 Detailed Description	2445
5.420.2 Member Typedef Documentation	2446
5.420.2.1 value_type	2446
5.420.3 Constructor & Destructor Documentation	2446
5.420.3.1 complex	2446
5.420.3.2 complex	2446
5.420.4 Member Function Documentation	2446
5.420.4.1 operator+=	2446
5.420.4.2 operator-=	2446
5.421std::condition_variable Class Reference	2448
5.421.1 Detailed Description	2448
5.422std::condition_variable_any Class Reference	2449
5.422.1 Detailed Description	2449
5.423std::conditional<_Cond, _Iftrue, _Iffalse> Struct Template Reference	2450
5.423.1 Detailed Description	2450
5.424std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg> Class Template Ref-	
erence	2451
5.424.1 Detailed Description	2451
5.424.2 Member Typedef Documentation	2452
5.424.2.1 first_argument_type	2452
5.424.2.2 result_type	2452
5.424.2.3 second_argument_type	2452
5.425std::const_mem_fun1_t<_Ret, _Tp, _Arg> Class Template Reference	2453
5.425.1 Detailed Description	2453

5.425.2 Member Typedef Documentation	2454
5.425.2.1 first_argument_type	2454
5.425.2.2 result_type	2454
5.425.2.3 second_argument_type	2454
5.426std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2455
5.426.1 Detailed Description	2455
5.426.2 Member Typedef Documentation	2456
5.426.2.1 argument_type	2456
5.426.2.2 result_type	2456
5.427std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	2457
5.427.1 Detailed Description	2457
5.427.2 Member Typedef Documentation	2458
5.427.2.1 argument_type	2458
5.427.2.2 result_type	2458
5.428std::ctype< _CharT > Class Template Reference	2459
5.428.1 Detailed Description	2461
5.428.2 Member Typedef Documentation	2462
5.428.2.1 char_type	2462
5.428.3 Member Function Documentation	2462
5.428.3.1 do_is	2462
5.428.3.2 do_is	2462
5.428.3.3 do_narrow	2463
5.428.3.4 do_narrow	2463
5.428.3.5 do_scan_is	2464
5.428.3.6 do_scan_not	2464
5.428.3.7 do_tolower	2465
5.428.3.8 do_tolower	2465
5.428.3.9 do_toupper	2466
5.428.3.10do_toupper	2466
5.428.3.11do_widen	2466
5.428.3.12do_widen	2467

5.428.3.13	is	2467
5.428.3.14	is	2468
5.428.3.15	narrow	2468
5.428.3.16	narrow	2469
5.428.3.17	scan_is	2469
5.428.3.18	scan_not	2470
5.428.3.19	tolower	2470
5.428.3.20	tolower	2471
5.428.3.21	toupper	2471
5.428.3.22	toupper	2471
5.428.3.23	widen	2472
5.428.3.24	widen	2472
5.428.4	Member Data Documentation	2473
5.428.4.1	id	2473
5.429	std::ctype< char > Class Template Reference	2474
5.429.1	Detailed Description	2476
5.429.2	Member Typedef Documentation	2476
5.429.2.1	char_type	2476
5.429.3	Constructor & Destructor Documentation	2477
5.429.3.1	ctype	2477
5.429.3.2	ctype	2477
5.429.3.3	~ctype	2477
5.429.4	Member Function Documentation	2477
5.429.4.1	classic_table	2477
5.429.4.2	do_narrow	2478
5.429.4.3	do_narrow	2478
5.429.4.4	do_tolower	2479
5.429.4.5	do_tolower	2479
5.429.4.6	do_toupper	2479
5.429.4.7	do_toupper	2480
5.429.4.8	do_widen	2480

5.429.4.9 do_widen	2481
5.429.4.10is	2481
5.429.4.11is	2481
5.429.4.12narrow	2482
5.429.4.13narrow	2482
5.429.4.14scan_is	2483
5.429.4.15scan_not	2483
5.429.4.16table	2484
5.429.4.17tolower	2484
5.429.4.18tolower	2484
5.429.4.19toupper	2485
5.429.4.20toupper	2485
5.429.4.21widen	2485
5.429.4.22widen	2486
5.429.5 Member Data Documentation	2486
5.429.5.1 id	2486
5.429.5.2 table_size	2487
5.430std::ctype< wchar_t > Class Template Reference	2488
5.430.1 Detailed Description	2490
5.430.2 Member Typedef Documentation	2491
5.430.2.1 char_type	2491
5.430.3 Constructor & Destructor Documentation	2491
5.430.3.1 ctype	2491
5.430.3.2 ctype	2491
5.430.3.3 ~ctype	2491
5.430.4 Member Function Documentation	2492
5.430.4.1 do_is	2492
5.430.4.2 do_is	2492
5.430.4.3 do_narrow	2493
5.430.4.4 do_narrow	2493
5.430.4.5 do_scan_is	2494

5.430.4.6 do_scan_not	2494
5.430.4.7 do_tolower	2495
5.430.4.8 do_tolower	2495
5.430.4.9 do_toupper	2495
5.430.4.10do_toupper	2496
5.430.4.11do_widen	2496
5.430.4.12do_widen	2497
5.430.4.13s	2497
5.430.4.14s	2498
5.430.4.15narrow	2498
5.430.4.16narrow	2499
5.430.4.17scan_is	2499
5.430.4.18scan_not	2499
5.430.4.19tolower	2500
5.430.4.20tolower	2500
5.430.4.21toupper	2501
5.430.4.22toupper	2501
5.430.4.23widen	2501
5.430.4.24widen	2502
5.430.5 Member Data Documentation	2502
5.430.5.1 id	2502
5.431std::ctype_base Struct Reference	2503
5.431.1 Detailed Description	2503
5.432std::ctype_byname<_CharT> Class Template Reference	2504
5.432.1 Detailed Description	2506
5.432.2 Member Typedef Documentation	2506
5.432.2.1 char_type	2506
5.432.3 Member Function Documentation	2507
5.432.3.1 do_is	2507
5.432.3.2 do_is	2507
5.432.3.3 do_narrow	2508

5.432.3.4	do_narrow	2508
5.432.3.5	do_scan_is	2509
5.432.3.6	do_scan_not	2509
5.432.3.7	do_tolower	2510
5.432.3.8	do_tolower	2510
5.432.3.9	do_toupper	2510
5.432.3.10	do_toupper	2511
5.432.3.11	do_widen	2511
5.432.3.12	do_widen	2512
5.432.3.13	is	2512
5.432.3.14	is	2513
5.432.3.15	narrow	2513
5.432.3.16	narrow	2514
5.432.3.17	scan_is	2514
5.432.3.18	scan_not	2515
5.432.3.19	tolower	2515
5.432.3.20	tolower	2516
5.432.3.21	toupper	2516
5.432.3.22	toupper	2516
5.432.3.23	widen	2517
5.432.3.24	widen	2517
5.432.4	Member Data Documentation	2518
5.432.4.1	id	2518
5.433	std::ctype_byname< char > Class Template Reference	2519
5.433.1	Detailed Description	2521
5.433.2	Member Typedef Documentation	2521
5.433.2.1	char_type	2521
5.433.3	Member Function Documentation	2522
5.433.3.1	classic_table	2522
5.433.3.2	do_narrow	2522
5.433.3.3	do_narrow	2522

5.433.3.4 do_tolower	2523
5.433.3.5 do_tolower	2523
5.433.3.6 do_toupper	2524
5.433.3.7 do_toupper	2524
5.433.3.8 do_widen	2524
5.433.3.9 do_widen	2525
5.433.3.10s	2525
5.433.3.11s	2526
5.433.3.12narrow	2526
5.433.3.13narrow	2527
5.433.3.14scan_is	2527
5.433.3.15scan_not	2527
5.433.3.16table	2528
5.433.3.17tolower	2528
5.433.3.18tolower	2528
5.433.3.19toupper	2529
5.433.3.20toupper	2529
5.433.3.21widen	2530
5.433.3.22widen	2530
5.433.4 Member Data Documentation	2531
5.433.4.1 id	2531
5.433.4.2 table_size	2531
5.434std::decay< _Tp > Class Template Reference	2532
5.434.1 Detailed Description	2532
5.435std::decimal::decimal128 Class Reference	2533
5.435.1 Detailed Description	2534
5.435.2 Constructor & Destructor Documentation	2534
5.435.2.1 decimal128	2534
5.436std::decimal::decimal32 Class Reference	2535
5.436.1 Detailed Description	2536
5.436.2 Constructor & Destructor Documentation	2536

5.436.2.1 decimal32	2536
5.437std::decimal::decimal64 Class Reference	2537
5.437.1 Detailed Description	2538
5.437.2 Constructor & Destructor Documentation	2538
5.437.2.1 decimal64	2538
5.438std::default_delete< _Tp > Struct Template Reference	2539
5.438.1 Detailed Description	2539
5.439std::default_delete< _Tp[]> Struct Template Reference	2540
5.439.1 Detailed Description	2540
5.440std::defer_lock_t Struct Reference	2541
5.440.1 Detailed Description	2541
5.441std::deque< _Tp, _Alloc > Class Template Reference	2542
5.441.1 Detailed Description	2546
5.441.2 Constructor & Destructor Documentation	2548
5.441.2.1 deque	2548
5.441.2.2 deque	2548
5.441.2.3 deque	2548
5.441.2.4 deque	2549
5.441.2.5 deque	2549
5.441.2.6 deque	2549
5.441.2.7 deque	2550
5.441.2.8 ~deque	2550
5.441.3 Member Function Documentation	2550
5.441.3.1 _M_fill_initialize	2550
5.441.3.2 _M_initialize_map	2551
5.441.3.3 _M_new_elements_at_back	2551
5.441.3.4 _M_new_elements_at_front	2552
5.441.3.5 _M_pop_back_aux	2552
5.441.3.6 _M_pop_front_aux	2552
5.441.3.7 _M_push_back_aux	2552
5.441.3.8 _M_push_front_aux	2553

5.441.3.9	<code>_M_range_check</code>	2553
5.441.3.10	<code>_M_range_initialize</code>	2553
5.441.3.11	<code>_M_range_initialize</code>	2553
5.441.3.12	<code>_M_reallocate_map</code>	2554
5.441.3.13	<code>_M_reserve_elements_at_back</code>	2554
5.441.3.14	<code>_M_reserve_elements_at_front</code>	2554
5.441.3.15	<code>_M_reserve_map_at_back</code>	2555
5.441.3.16	<code>_M_reserve_map_at_front</code>	2555
5.441.3.17	<code>assign</code>	2555
5.441.3.18	<code>assign</code>	2556
5.441.3.19	<code>assign</code>	2556
5.441.3.20	<code>at</code>	2556
5.441.3.21	<code>at</code>	2557
5.441.3.22	<code>back</code>	2557
5.441.3.23	<code>back</code>	2558
5.441.3.24	<code>begin</code>	2558
5.441.3.25	<code>begin</code>	2558
5.441.3.26	<code>begin</code>	2558
5.441.3.27	<code>end</code>	2558
5.441.3.28	<code>clear</code>	2559
5.441.3.29	<code>begin</code>	2559
5.441.3.30	<code>end</code>	2559
5.441.3.31	<code>emplace</code>	2559
5.441.3.32	<code>empty</code>	2560
5.441.3.33	<code>end</code>	2560
5.441.3.34	<code>end</code>	2560
5.441.3.35	<code>erase</code>	2560
5.441.3.36	<code>erase</code>	2561
5.441.3.37	<code>front</code>	2561
5.441.3.38	<code>front</code>	2562
5.441.3.39	<code>get_allocator</code>	2562

5.441.3.40	insert	2562
5.441.3.41	insert	2562
5.441.3.42	insert	2563
5.441.3.43	insert	2563
5.441.3.44	insert	2564
5.441.3.45	max_size	2564
5.441.3.46	operator=	2564
5.441.3.47	operator=	2565
5.441.3.48	operator=	2565
5.441.3.49	operator[]	2565
5.441.3.50	operator[]	2566
5.441.3.51	pop_back	2566
5.441.3.52	pop_front	2566
5.441.3.53	push_back	2567
5.441.3.54	push_front	2567
5.441.3.55	begin	2568
5.441.3.56	begin	2568
5.441.3.57	end	2568
5.441.3.58	end	2568
5.441.3.59	resize	2568
5.441.3.60	shrink_to_fit	2569
5.441.3.61	size	2569
5.441.3.62	swap	2569
5.442	std::discard_block_engine< _RandomNumberEngine, __p, __r >	
	Class Template Reference	2571
5.442.1	Detailed Description	2572
5.442.2	Member Typedef Documentation	2572
5.442.2.1	result_type	2572
5.442.3	Constructor & Destructor Documentation	2572
5.442.3.1	discard_block_engine	2572
5.442.3.2	discard_block_engine	2573

5.442.3.3	discard_block_engine	2573
5.442.3.4	discard_block_engine	2573
5.442.3.5	discard_block_engine	2574
5.442.4	Member Function Documentation	2574
5.442.4.1	base	2574
5.442.4.2	discard	2574
5.442.4.3	max	2574
5.442.4.4	min	2575
5.442.4.5	operator()	2575
5.442.4.6	seed	2575
5.442.4.7	seed	2575
5.442.4.8	seed	2576
5.442.5	Friends And Related Function Documentation	2576
5.442.5.1	operator<<	2576
5.442.5.2	operator==	2576
5.442.5.3	operator>>	2577
5.443	std::discrete_distribution<_IntType> Class Template Reference	2578
5.443.1	Detailed Description	2579
5.443.2	Member Typedef Documentation	2579
5.443.2.1	result_type	2579
5.443.3	Member Function Documentation	2579
5.443.3.1	max	2579
5.443.3.2	min	2579
5.443.3.3	param	2579
5.443.3.4	param	2580
5.443.3.5	probabilities	2580
5.443.3.6	reset	2580
5.443.4	Friends And Related Function Documentation	2580
5.443.4.1	operator<<	2580
5.443.4.2	operator>>	2581
5.444	std::discrete_distribution<_IntType>::param_type Struct Reference	2582

5.444.1 Detailed Description	2582
5.445std::divides< _Tp > Struct Template Reference	2583
5.445.1 Detailed Description	2583
5.445.2 Member Typedef Documentation	2584
5.445.2.1 first_argument_type	2584
5.445.2.2 result_type	2584
5.445.2.3 second_argument_type	2584
5.446std::domain_error Class Reference	2585
5.446.1 Detailed Description	2585
5.446.2 Member Function Documentation	2585
5.446.2.1 what	2585
5.447std::enable_if< bool, _Tp > Struct Template Reference	2587
5.447.1 Detailed Description	2587
5.448std::enable_shared_from_this< _Tp > Class Template Reference	2588
5.448.1 Detailed Description	2588
5.449std::equal_to< _Tp > Struct Template Reference	2589
5.449.1 Detailed Description	2589
5.449.2 Member Typedef Documentation	2590
5.449.2.1 first_argument_type	2590
5.449.2.2 result_type	2590
5.449.2.3 second_argument_type	2590
5.450std::error_category Class Reference	2591
5.450.1 Detailed Description	2591
5.451std::error_code Struct Reference	2592
5.451.1 Detailed Description	2592
5.452std::error_condition Struct Reference	2593
5.452.1 Detailed Description	2593
5.453std::exception Class Reference	2595
5.453.1 Detailed Description	2596
5.453.2 Member Function Documentation	2596
5.453.2.1 what	2596

5.454	std::exponential_distribution< _RealType > Class Template Reference	2597
5.454.1	Detailed Description	2597
5.454.2	Member Typedef Documentation	2598
5.454.2.1	result_type	2598
5.454.3	Constructor & Destructor Documentation	2598
5.454.3.1	exponential_distribution	2598
5.454.4	Member Function Documentation	2598
5.454.4.1	lambda	2598
5.454.4.2	max	2598
5.454.4.3	min	2599
5.454.4.4	param	2599
5.454.4.5	param	2599
5.454.4.6	reset	2599
5.455	std::exponential_distribution< _RealType >::param_type Struct Reference	2600
5.455.1	Detailed Description	2600
5.456	std::extent< typename, _Uint > Struct Template Reference	2601
5.456.1	Detailed Description	2601
5.457	std::extreme_value_distribution< _RealType > Class Template Reference	2602
5.457.1	Detailed Description	2602
5.457.2	Member Typedef Documentation	2603
5.457.2.1	result_type	2603
5.457.3	Member Function Documentation	2603
5.457.3.1	a	2603
5.457.3.2	b	2603
5.457.3.3	max	2603
5.457.3.4	min	2603
5.457.3.5	param	2604
5.457.3.6	param	2604
5.457.3.7	reset	2604

5.458	<code>std::extreme_value_distribution< _RealType >::param_type</code> Struct Reference	2605
5.458.1	Detailed Description	2605
5.459	<code>std::fisher_f_distribution< _RealType ></code> Class Template Reference	2606
5.459.1	Detailed Description	2607
5.459.2	Member Typedef Documentation	2607
5.459.2.1	<code>result_type</code>	2607
5.459.3	Member Function Documentation	2607
5.459.3.1	<code>max</code>	2607
5.459.3.2	<code>min</code>	2607
5.459.3.3	<code>param</code>	2607
5.459.3.4	<code>param</code>	2608
5.459.3.5	<code>reset</code>	2608
5.459.4	Friends And Related Function Documentation	2608
5.459.4.1	<code>operator<<</code>	2608
5.459.4.2	<code>operator>></code>	2609
5.460	<code>std::fisher_f_distribution< _RealType >::param_type</code> Struct Reference	2610
5.460.1	Detailed Description	2610
5.461	<code>std::forward_iterator_tag</code> Struct Reference	2611
5.461.1	Detailed Description	2611
5.462	<code>std::forward_list< _Tp, _Alloc ></code> Class Template Reference	2612
5.462.1	Detailed Description	2615
5.462.2	Constructor & Destructor Documentation	2615
5.462.2.1	<code>forward_list</code>	2615
5.462.2.2	<code>forward_list</code>	2615
5.462.2.3	<code>forward_list</code>	2616
5.462.2.4	<code>forward_list</code>	2616
5.462.2.5	<code>forward_list</code>	2616
5.462.2.6	<code>forward_list</code>	2617
5.462.2.7	<code>forward_list</code>	2617
5.462.2.8	<code>forward_list</code>	2617

5.462.2.9 forward_list	2618
5.462.2.10 ~forward_list	2618
5.462.3 Member Function Documentation	2618
5.462.3.1 assign	2618
5.462.3.2 assign	2619
5.462.3.3 assign	2619
5.462.3.4 before_begin	2619
5.462.3.5 before_begin	2620
5.462.3.6 begin	2620
5.462.3.7 begin	2620
5.462.3.8 cbefore_begin	2620
5.462.3.9 cbegin	2621
5.462.3.10 cend	2621
5.462.3.11 clear	2621
5.462.3.12 emplace_after	2621
5.462.3.13 emplace_front	2622
5.462.3.14 empty	2622
5.462.3.15 end	2622
5.462.3.16 end	2622
5.462.3.17 erase_after	2623
5.462.3.18 erase_after	2623
5.462.3.19 front	2624
5.462.3.20 front	2624
5.462.3.21 get_allocator	2624
5.462.3.22 insert_after	2624
5.462.3.23 insert_after	2625
5.462.3.24 insert_after	2625
5.462.3.25 insert_after	2626
5.462.3.26 max_size	2626
5.462.3.27 merge	2626
5.462.3.28 merge	2627

5.462.3.29	operator=	2627
5.462.3.30	operator=	2628
5.462.3.31	operator=	2628
5.462.3.32	pop_front	2628
5.462.3.33	push_front	2629
5.462.3.34	remove	2629
5.462.3.35	remove_if	2629
5.462.3.36	resize	2630
5.462.3.37	resize	2630
5.462.3.38	reverse	2630
5.462.3.39	sort	2631
5.462.3.40	sort	2631
5.462.3.41	splice_after	2631
5.462.3.42	splice_after	2631
5.462.3.43	splice_after	2632
5.462.3.44	swap	2632
5.462.3.45	unique	2633
5.462.3.46	unique	2633
5.463	std::fpos< _StateT > Class Template Reference	2634
5.463.1	Detailed Description	2634
5.463.2	Constructor & Destructor Documentation	2634
5.463.2.1	fpos	2634
5.463.3	Member Function Documentation	2635
5.463.3.1	operator streamoff	2635
5.463.3.2	operator+	2635
5.463.3.3	operator+=	2635
5.463.3.4	operator-	2635
5.463.3.5	operator-	2635
5.463.3.6	operator-=	2635
5.463.3.7	state	2636
5.463.3.8	state	2636

5.464std::front_insert_iterator< _Container > Class Template Reference	2637
5.464.1 Detailed Description	2638
5.464.2 Member Typedef Documentation	2638
5.464.2.1 container_type	2638
5.464.2.2 difference_type	2638
5.464.2.3 iterator_category	2638
5.464.2.4 pointer	2638
5.464.2.5 reference	2639
5.464.2.6 value_type	2639
5.464.3 Constructor & Destructor Documentation	2639
5.464.3.1 front_insert_iterator	2639
5.464.4 Member Function Documentation	2639
5.464.4.1 operator*	2639
5.464.4.2 operator++	2639
5.464.4.3 operator++	2640
5.464.4.4 operator=	2640
5.465std::function< _Res(_ArgTypes...)> Class Template Reference	2641
5.465.1 Detailed Description	2642
5.465.2 Constructor & Destructor Documentation	2643
5.465.2.1 function	2643
5.465.2.2 function	2643
5.465.2.3 function	2643
5.465.2.4 function	2644
5.465.2.5 function	2644
5.465.3 Member Function Documentation	2644
5.465.3.1 operator bool	2644
5.465.3.2 operator()	2645
5.465.3.3 operator=	2645
5.465.3.4 operator=	2645
5.465.3.5 operator=	2646
5.465.3.6 operator=	2646

5.465.3.7 operator=	2647
5.465.3.8 swap	2647
5.465.3.9 target	2647
5.465.3.10target	2648
5.465.3.11target_type	2648
5.466std::future< _Res > Class Template Reference	2649
5.466.1 Detailed Description	2650
5.466.2 Constructor & Destructor Documentation	2650
5.466.2.1 future	2650
5.466.3 Member Function Documentation	2650
5.466.3.1 _M_get_result	2650
5.466.3.2 get	2650
5.467std::future< _Res & > Class Template Reference	2652
5.467.1 Detailed Description	2653
5.467.2 Constructor & Destructor Documentation	2653
5.467.2.1 future	2653
5.467.3 Member Function Documentation	2653
5.467.3.1 _M_get_result	2653
5.467.3.2 get	2654
5.468std::future< void > Class Template Reference	2655
5.468.1 Detailed Description	2656
5.468.2 Constructor & Destructor Documentation	2656
5.468.2.1 future	2656
5.468.3 Member Function Documentation	2656
5.468.3.1 _M_get_result	2656
5.468.3.2 get	2656
5.469std::future_error Class Reference	2658
5.469.1 Detailed Description	2658
5.469.2 Member Function Documentation	2658
5.469.2.1 what	2658
5.470std::gamma_distribution< _RealType > Class Template Reference	2660

5.470.1 Detailed Description	2661
5.470.2 Member Typedef Documentation	2661
5.470.2.1 result_type	2661
5.470.3 Constructor & Destructor Documentation	2661
5.470.3.1 gamma_distribution	2661
5.470.4 Member Function Documentation	2662
5.470.4.1 alpha	2662
5.470.4.2 beta	2662
5.470.4.3 max	2662
5.470.4.4 min	2662
5.470.4.5 operator()	2662
5.470.4.6 param	2663
5.470.4.7 param	2663
5.470.4.8 reset	2663
5.470.5 Friends And Related Function Documentation	2663
5.470.5.1 operator<<	2663
5.470.5.2 operator>>	2664
5.471 std::gamma_distribution< _RealType >::param_type Struct Reference	2665
5.471.1 Detailed Description	2665
5.472 std::geometric_distribution< _IntType > Class Template Reference .	2666
5.472.1 Detailed Description	2666
5.472.2 Member Typedef Documentation	2667
5.472.2.1 result_type	2667
5.472.3 Member Function Documentation	2667
5.472.3.1 max	2667
5.472.3.2 min	2667
5.472.3.3 p	2667
5.472.3.4 param	2667
5.472.3.5 param	2668
5.472.3.6 reset	2668
5.473 std::geometric_distribution< _IntType >::param_type Struct Reference	2669

5.473.1 Detailed Description	2669
5.474std::greater< _Tp > Struct Template Reference	2670
5.474.1 Detailed Description	2670
5.474.2 Member Typedef Documentation	2671
5.474.2.1 first_argument_type	2671
5.474.2.2 result_type	2671
5.474.2.3 second_argument_type	2671
5.475std::greater_equal< _Tp > Struct Template Reference	2672
5.475.1 Detailed Description	2672
5.475.2 Member Typedef Documentation	2673
5.475.2.1 first_argument_type	2673
5.475.2.2 result_type	2673
5.475.2.3 second_argument_type	2673
5.476std::gslice Class Reference	2674
5.476.1 Detailed Description	2674
5.477std::gslice_array< _Tp > Class Template Reference	2675
5.477.1 Detailed Description	2676
5.478std::has_nothrow_assign< _Tp > Struct Template Reference	2677
5.478.1 Detailed Description	2677
5.479std::has_nothrow_copy_constructor< _Tp > Struct Template Reference	2678
5.479.1 Detailed Description	2678
5.480std::has_nothrow_default_constructor< _Tp > Struct Template Refer-	
ence	2679
5.480.1 Detailed Description	2679
5.481std::has_trivial_assign< _Tp > Struct Template Reference	2680
5.481.1 Detailed Description	2680
5.482std::has_trivial_copy_constructor< _Tp > Struct Template Reference	2681
5.482.1 Detailed Description	2681
5.483std::has_trivial_default_constructor< _Tp > Struct Template Reference	2682
5.483.1 Detailed Description	2682
5.484std::has_trivial_destructor< _Tp > Struct Template Reference	2683

5.484.1 Detailed Description	2683
5.485std::has_virtual_destructor< _Tp > Struct Template Reference	2684
5.485.1 Detailed Description	2684
5.486std::hash< _Tp > Struct Template Reference	2685
5.486.1 Detailed Description	2686
5.486.2 Member Typedef Documentation	2686
5.486.2.1 argument_type	2686
5.486.2.2 result_type	2686
5.487std::hash< ::bitset< _Nb > > Struct Template Reference	2688
5.487.1 Detailed Description	2688
5.487.2 Member Typedef Documentation	2689
5.487.2.1 argument_type	2689
5.487.2.2 result_type	2689
5.488std::hash< ::vector< bool, _Alloc > > Struct Template Reference . .	2690
5.488.1 Detailed Description	2690
5.488.2 Member Typedef Documentation	2691
5.488.2.1 argument_type	2691
5.488.2.2 result_type	2691
5.489std::hash< __debug::bitset< _Nb > > Struct Template Reference . .	2692
5.489.1 Detailed Description	2692
5.489.2 Member Typedef Documentation	2693
5.489.2.1 argument_type	2693
5.489.2.2 result_type	2693
5.490std::hash< __debug::vector< bool, _Alloc > > Struct Template Ref- erence	2694
5.490.1 Detailed Description	2694
5.490.2 Member Typedef Documentation	2695
5.490.2.1 argument_type	2695
5.490.2.2 result_type	2695
5.491std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	2696
5.491.1 Detailed Description	2696

5.491.2 Member Typedef Documentation	2697
5.491.2.1 argument_type	2697
5.491.2.2 result_type	2697
5.492std::hash< __gnu_cxx::throw_value_random > Struct Template Reference	2698
5.492.1 Detailed Description	2698
5.492.2 Member Typedef Documentation	2699
5.492.2.1 argument_type	2699
5.492.2.2 result_type	2699
5.493std::hash< __profile::bitset< _Nb > > Struct Template Reference	2700
5.493.1 Detailed Description	2700
5.493.2 Member Typedef Documentation	2701
5.493.2.1 argument_type	2701
5.493.2.2 result_type	2701
5.494std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference	2702
5.494.1 Detailed Description	2702
5.494.2 Member Typedef Documentation	2703
5.494.2.1 argument_type	2703
5.494.2.2 result_type	2703
5.495std::hash< _Tp * > Struct Template Reference	2704
5.495.1 Detailed Description	2704
5.495.2 Member Typedef Documentation	2705
5.495.2.1 argument_type	2705
5.495.2.2 result_type	2705
5.496std::hash< error_code > Struct Template Reference	2706
5.496.1 Detailed Description	2706
5.496.2 Member Typedef Documentation	2707
5.496.2.1 argument_type	2707
5.496.2.2 result_type	2707
5.497std::hash< string > Struct Template Reference	2708
5.497.1 Detailed Description	2708

5.497.2 Member Typedef Documentation	2709
5.497.2.1 argument_type	2709
5.497.2.2 result_type	2709
5.498std::hash< thread::id > Struct Template Reference	2710
5.498.1 Detailed Description	2710
5.498.2 Member Typedef Documentation	2711
5.498.2.1 argument_type	2711
5.498.2.2 result_type	2711
5.499std::hash< u16string > Struct Template Reference	2712
5.499.1 Detailed Description	2712
5.499.2 Member Typedef Documentation	2713
5.499.2.1 argument_type	2713
5.499.2.2 result_type	2713
5.500std::hash< u32string > Struct Template Reference	2714
5.500.1 Detailed Description	2714
5.500.2 Member Typedef Documentation	2715
5.500.2.1 argument_type	2715
5.500.2.2 result_type	2715
5.501std::hash< wstring > Struct Template Reference	2716
5.501.1 Detailed Description	2716
5.501.2 Member Typedef Documentation	2717
5.501.2.1 argument_type	2717
5.501.2.2 result_type	2717
5.502std::identity< _Tp > Struct Template Reference	2718
5.502.1 Detailed Description	2718
5.503std::independent_bits_engine< _RandomNumberEngine, __w, _- UIntType > Class Template Reference	2719
5.503.1 Detailed Description	2720
5.503.2 Member Typedef Documentation	2720
5.503.2.1 result_type	2720
5.503.3 Constructor & Destructor Documentation	2720

5.503.3.1 independent_bits_engine	2720
5.503.3.2 independent_bits_engine	2720
5.503.3.3 independent_bits_engine	2721
5.503.3.4 independent_bits_engine	2721
5.503.3.5 independent_bits_engine	2721
5.503.4 Member Function Documentation	2722
5.503.4.1 base	2722
5.503.4.2 discard	2722
5.503.4.3 max	2722
5.503.4.4 min	2722
5.503.4.5 operator()	2723
5.503.4.6 seed	2723
5.503.4.7 seed	2723
5.503.4.8 seed	2723
5.503.5 Friends And Related Function Documentation	2724
5.503.5.1 operator==	2724
5.503.5.2 operator>>	2724
5.504std::indirect_array< _Tp > Class Template Reference	2725
5.504.1 Detailed Description	2726
5.505std::initializer_list< _E > Class Template Reference	2727
5.505.1 Detailed Description	2727
5.506std::input_iterator_tag Struct Reference	2728
5.506.1 Detailed Description	2728
5.507std::insert_iterator< _Container > Class Template Reference	2729
5.507.1 Detailed Description	2730
5.507.2 Member Typedef Documentation	2730
5.507.2.1 container_type	2730
5.507.2.2 difference_type	2730
5.507.2.3 iterator_category	2730
5.507.2.4 pointer	2730
5.507.2.5 reference	2731

5.507.2.6 value_type	2731
5.507.3 Constructor & Destructor Documentation	2731
5.507.3.1 insert_iterator	2731
5.507.4 Member Function Documentation	2731
5.507.4.1 operator*	2731
5.507.4.2 operator++	2731
5.507.4.3 operator++	2732
5.507.4.4 operator=	2732
5.508std::integral_constant< _Tp, __v > Struct Template Reference	2733
5.508.1 Detailed Description	2733
5.509std::invalid_argument Class Reference	2734
5.509.1 Detailed Description	2734
5.509.2 Member Function Documentation	2734
5.509.2.1 what	2734
5.510std::ios_base Class Reference	2736
5.510.1 Detailed Description	2739
5.510.2 Member Typedef Documentation	2739
5.510.2.1 event_callback	2739
5.510.2.2 fmtflags	2739
5.510.2.3 iostate	2740
5.510.2.4 openmode	2740
5.510.2.5 seekdir	2741
5.510.3 Member Enumeration Documentation	2741
5.510.3.1 event	2741
5.510.4 Constructor & Destructor Documentation	2741
5.510.4.1 ~ios_base	2741
5.510.5 Member Function Documentation	2741
5.510.5.1 _M_getloc	2741
5.510.5.2 flags	2742
5.510.5.3 flags	2742
5.510.5.4 getloc	2742

5.510.5.5 imbue	2743
5.510.5.6 iword	2743
5.510.5.7 precision	2744
5.510.5.8 precision	2744
5.510.5.9 pword	2744
5.510.5.10register_callback	2745
5.510.5.11setf	2745
5.510.5.12setf	2745
5.510.5.13sync_with_stdio	2746
5.510.5.14unsetf	2746
5.510.5.15width	2746
5.510.5.16width	2747
5.510.5.17xalloc	2747
5.510.6 Member Data Documentation	2747
5.510.6.1 adjustfield	2747
5.510.6.2 app	2748
5.510.6.3 ate	2748
5.510.6.4 badbit	2748
5.510.6.5 basefield	2748
5.510.6.6 beg	2749
5.510.6.7 binary	2749
5.510.6.8 boolalpha	2749
5.510.6.9 cur	2749
5.510.6.10dec	2749
5.510.6.11lend	2749
5.510.6.12eofbit	2750
5.510.6.13failbit	2750
5.510.6.14fixed	2750
5.510.6.15floatfield	2750
5.510.6.16goodbit	2751
5.510.6.17hex	2751

5.510.6.18n	2751
5.510.6.19internal	2752
5.510.6.20left	2752
5.510.6.21oct	2752
5.510.6.22out	2752
5.510.6.23right	2752
5.510.6.24scientific	2752
5.510.6.25showbase	2753
5.510.6.26showpoint	2753
5.510.6.27showpos	2753
5.510.6.28skipws	2753
5.510.6.29trunc	2753
5.510.6.30unitbuf	2753
5.510.6.31uppercase	2753
5.511std::ios_base::failure Class Reference	2755
5.511.1 Detailed Description	2755
5.511.2 Member Function Documentation	2755
5.511.2.1 what	2755
5.512std::is_abstract< _Tp > Struct Template Reference	2756
5.512.1 Detailed Description	2756
5.513std::is_arithmetic< _Tp > Struct Template Reference	2757
5.513.1 Detailed Description	2757
5.514std::is_array< typename > Struct Template Reference	2758
5.514.1 Detailed Description	2758
5.515std::is_base_of< _Base, _Derived > Struct Template Reference	2759
5.515.1 Detailed Description	2759
5.516std::is_bind_expression< _Tp > Struct Template Reference	2760
5.516.1 Detailed Description	2760
5.517std::is_bind_expression< _Bind< _Signature > > Struct Template Reference	2762
5.517.1 Detailed Description	2762

5.518	<code>std::is_bind_expression< _Bind_result< _Result, _Signature > ></code>	
	Struct Template Reference	2763
5.518.1	Detailed Description	2763
5.519	<code>std::is_class< _Tp ></code>	Struct Template Reference
		2764
5.519.1	Detailed Description	2764
5.520	<code>std::is_compound< _Tp ></code>	Struct Template Reference
		2765
5.520.1	Detailed Description	2765
5.521	<code>std::is_const< typename ></code>	Struct Template Reference
		2766
5.521.1	Detailed Description	2766
5.522	<code>std::is_constructible< _Tp, _Args ></code>	Struct Template Reference
		2767
5.522.1	Detailed Description	2767
5.523	<code>std::is_convertible< _From, _To ></code>	Struct Template Reference
		2768
5.523.1	Detailed Description	2768
5.524	<code>std::is_empty< _Tp ></code>	Struct Template Reference
		2769
5.524.1	Detailed Description	2769
5.525	<code>std::is_enum< _Tp ></code>	Struct Template Reference
		2770
5.525.1	Detailed Description	2770
5.526	<code>std::is_error_code_enum< _Tp ></code>	Struct Template Reference
		2771
5.526.1	Detailed Description	2771
5.527	<code>std::is_error_condition_enum< _Tp ></code>	Struct Template Reference
		2772
5.527.1	Detailed Description	2772
5.528	<code>std::is_explicitly_convertible< _From, _To ></code>	Struct Template Reference
		2773
5.528.1	Detailed Description	2773
5.529	<code>std::is_floating_point< _Tp ></code>	Struct Template Reference
		2774
5.529.1	Detailed Description	2774
5.530	<code>std::is_function< typename ></code>	Struct Template Reference
		2775
5.530.1	Detailed Description	2775
5.531	<code>std::is_fundamental< _Tp ></code>	Struct Template Reference
		2776
5.531.1	Detailed Description	2776
5.532	<code>std::is_integral< _Tp ></code>	Struct Template Reference
		2777
5.532.1	Detailed Description	2777

5.533std::is_lvalue_reference< typename > Struct Template Reference . . .	2778
5.533.1 Detailed Description	2778
5.534std::is_member_function_pointer< _Tp > Struct Template Reference	2779
5.534.1 Detailed Description	2779
5.535std::is_member_object_pointer< _Tp > Struct Template Reference .	2780
5.535.1 Detailed Description	2780
5.536std::is_object< _Tp > Struct Template Reference	2781
5.536.1 Detailed Description	2781
5.537std::is_placeholder< _Tp > Struct Template Reference	2782
5.537.1 Detailed Description	2782
5.538std::is_placeholder< _Placeholder< _Num > > Struct Template Reference	2784
5.538.1 Detailed Description	2784
5.539std::is_pod< _Tp > Struct Template Reference	2785
5.539.1 Detailed Description	2785
5.540std::is_pointer< _Tp > Struct Template Reference	2786
5.540.1 Detailed Description	2786
5.541std::is_polymorphic< _Tp > Struct Template Reference	2787
5.541.1 Detailed Description	2787
5.542std::is_reference< _Tp > Struct Template Reference	2788
5.542.1 Detailed Description	2788
5.543std::is_rvalue_reference< typename > Struct Template Reference . . .	2789
5.543.1 Detailed Description	2789
5.544std::is_same< typename, typename > Struct Template Reference . . .	2790
5.544.1 Detailed Description	2790
5.545std::is_scalar< _Tp > Struct Template Reference	2791
5.545.1 Detailed Description	2791
5.546std::is_signed< _Tp > Struct Template Reference	2792
5.546.1 Detailed Description	2792
5.547std::is_standard_layout< _Tp > Struct Template Reference	2793
5.547.1 Detailed Description	2793

5.548	<code>std::is_trivial< _Tp ></code> Struct Template Reference	2794
5.548.1	Detailed Description	2794
5.549	<code>std::is_union< _Tp ></code> Struct Template Reference	2795
5.549.1	Detailed Description	2795
5.550	<code>std::is_unsigned< _Tp ></code> Struct Template Reference	2796
5.550.1	Detailed Description	2796
5.551	<code>std::is_void< _Tp ></code> Struct Template Reference	2797
5.551.1	Detailed Description	2797
5.552	<code>std::is_volatile< typename ></code> Struct Template Reference	2798
5.552.1	Detailed Description	2798
5.553	<code>std::istream_iterator< _Tp, _CharT, _Traits, _Dist ></code> Class Template Reference	2799
5.553.1	Detailed Description	2800
5.553.2	Member Typedef Documentation	2800
5.553.2.1	<code>difference_type</code>	2800
5.553.2.2	<code>iterator_category</code>	2800
5.553.2.3	<code>pointer</code>	2800
5.553.2.4	<code>reference</code>	2800
5.553.2.5	<code>value_type</code>	2800
5.553.3	Constructor & Destructor Documentation	2801
5.553.3.1	<code>istream_iterator</code>	2801
5.553.3.2	<code>istream_iterator</code>	2801
5.554	<code>std::istreambuf_iterator< _CharT, _Traits ></code> Class Template Reference	2802
5.554.1	Detailed Description	2803
5.554.2	Member Typedef Documentation	2803
5.554.2.1	<code>char_type</code>	2803
5.554.2.2	<code>difference_type</code>	2803
5.554.2.3	<code>int_type</code>	2803
5.554.2.4	<code>istream_type</code>	2803
5.554.2.5	<code>iterator_category</code>	2804
5.554.2.6	<code>pointer</code>	2804

5.554.2.7 reference	2804
5.554.2.8 streambuf_type	2804
5.554.2.9 traits_type	2804
5.554.2.10value_type	2805
5.554.3 Constructor & Destructor Documentation	2805
5.554.3.1 istreambuf_iterator	2805
5.554.3.2 istreambuf_iterator	2805
5.554.3.3 istreambuf_iterator	2805
5.554.4 Member Function Documentation	2805
5.554.4.1 equal	2805
5.554.4.2 operator*	2806
5.554.4.3 operator++	2806
5.554.4.4 operator++	2806
5.555std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	2807
5.555.1 Detailed Description	2807
5.555.2 Member Typedef Documentation	2807
5.555.2.1 difference_type	2807
5.555.2.2 iterator_category	2808
5.555.2.3 pointer	2808
5.555.2.4 reference	2808
5.555.2.5 value_type	2808
5.556std::iterator_traits< _Iterator > Struct Template Reference	2809
5.556.1 Detailed Description	2809
5.557std::iterator_traits< _Tp * > Struct Template Reference	2810
5.557.1 Detailed Description	2810
5.558std::iterator_traits< const _Tp * > Struct Template Reference	2811
5.558.1 Detailed Description	2811
5.559std::length_error Class Reference	2812
5.559.1 Detailed Description	2812
5.559.2 Member Function Documentation	2812

5.559.2.1 what	2812
5.560std::less< _Tp > Struct Template Reference	2814
5.560.1 Detailed Description	2814
5.560.2 Member Typedef Documentation	2815
5.560.2.1 first_argument_type	2815
5.560.2.2 result_type	2815
5.560.2.3 second_argument_type	2815
5.561std::less_equal< _Tp > Struct Template Reference	2816
5.561.1 Detailed Description	2816
5.561.2 Member Typedef Documentation	2817
5.561.2.1 first_argument_type	2817
5.561.2.2 result_type	2817
5.561.2.3 second_argument_type	2817
5.562std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2818
5.562.1 Detailed Description	2819
5.562.2 Member Typedef Documentation	2819
5.562.2.1 result_type	2819
5.562.3 Constructor & Destructor Documentation	2819
5.562.3.1 linear_congruential_engine	2819
5.562.3.2 linear_congruential_engine	2820
5.562.4 Member Function Documentation	2820
5.562.4.1 discard	2820
5.562.4.2 max	2820
5.562.4.3 min	2821
5.562.4.4 operator()	2821
5.562.4.5 seed	2821
5.562.4.6 seed	2822
5.562.5 Friends And Related Function Documentation	2822
5.562.5.1 operator<<	2822
5.562.5.2 operator==	2823

5.562.5.3 operator>>	2823
5.562.6 Member Data Documentation	2824
5.562.6.1 increment	2824
5.562.6.2 modulus	2824
5.562.6.3 multiplier	2824
5.563std::list<_Tp, _Alloc> Class Template Reference	2825
5.563.1 Detailed Description	2828
5.563.2 Constructor & Destructor Documentation	2829
5.563.2.1 list	2829
5.563.2.2 list	2829
5.563.2.3 list	2830
5.563.2.4 list	2830
5.563.2.5 list	2830
5.563.2.6 list	2831
5.563.2.7 list	2831
5.563.3 Member Function Documentation	2831
5.563.3.1 _M_create_node	2831
5.563.3.2 assign	2832
5.563.3.3 assign	2832
5.563.3.4 assign	2832
5.563.3.5 back	2833
5.563.3.6 back	2833
5.563.3.7 begin	2833
5.563.3.8 begin	2833
5.563.3.9 cbegin	2834
5.563.3.10end	2834
5.563.3.11clear	2834
5.563.3.12rbegin	2834
5.563.3.13rend	2834
5.563.3.14emplace	2835
5.563.3.15empty	2835

5.563.3.16end	2835
5.563.3.17end	2835
5.563.3.18erase	2836
5.563.3.19erase	2836
5.563.3.20front	2837
5.563.3.21front	2837
5.563.3.22get_allocator	2837
5.563.3.23insert	2837
5.563.3.24insert	2838
5.563.3.25insert	2838
5.563.3.26insert	2839
5.563.3.27insert	2839
5.563.3.28max_size	2840
5.563.3.29merge	2840
5.563.3.30merge	2840
5.563.3.31operator=	2841
5.563.3.32operator=	2841
5.563.3.33operator=	2841
5.563.3.34pop_back	2842
5.563.3.35pop_front	2842
5.563.3.36push_back	2842
5.563.3.37push_front	2843
5.563.3.38begin	2843
5.563.3.39begin	2843
5.563.3.40remove	2843
5.563.3.41remove_if	2844
5.563.3.42end	2844
5.563.3.43end	2844
5.563.3.44resize	2844
5.563.3.45reverse	2845
5.563.3.46size	2845

5.563.3.47	sort	2845
5.563.3.48	sort	2846
5.563.3.49	splice	2846
5.563.3.50	splice	2846
5.563.3.51	splice	2847
5.563.3.52	swap	2847
5.563.3.53	unique	2847
5.563.3.54	unique	2848
5.564	std::locale Class Reference	2849
5.564.1	Detailed Description	2850
5.564.2	Member Typedef Documentation	2851
5.564.2.1	category	2851
5.564.3	Constructor & Destructor Documentation	2851
5.564.3.1	locale	2851
5.564.3.2	locale	2851
5.564.3.3	locale	2851
5.564.3.4	locale	2851
5.564.3.5	locale	2852
5.564.3.6	locale	2852
5.564.3.7	~locale	2852
5.564.4	Member Function Documentation	2853
5.564.4.1	classic	2853
5.564.4.2	combine	2853
5.564.4.3	global	2853
5.564.4.4	name	2854
5.564.4.5	operator!=	2854
5.564.4.6	operator()	2854
5.564.4.7	operator=	2855
5.564.4.8	operator==	2855
5.564.5	Member Data Documentation	2855
5.564.5.1	all	2855

5.564.5.2 collate	2855
5.564.5.3 ctype	2856
5.564.5.4 messages	2856
5.564.5.5 monetary	2856
5.564.5.6 none	2856
5.564.5.7 numeric	2856
5.564.5.8 time	2857
5.565std::locale::facet Class Reference	2858
5.565.1 Detailed Description	2859
5.565.2 Constructor & Destructor Documentation	2859
5.565.2.1 facet	2859
5.565.2.2 ~facet	2859
5.566std::locale::id Class Reference	2860
5.566.1 Detailed Description	2860
5.566.2 Constructor & Destructor Documentation	2860
5.566.2.1 id	2860
5.566.3 Friends And Related Function Documentation	2861
5.566.3.1 has_facet	2861
5.566.3.2 use_facet	2861
5.567std::lock_guard< _Mutex > Class Template Reference	2862
5.567.1 Detailed Description	2862
5.568std::logic_error Class Reference	2863
5.568.1 Detailed Description	2863
5.568.2 Constructor & Destructor Documentation	2863
5.568.2.1 logic_error	2863
5.568.3 Member Function Documentation	2864
5.568.3.1 what	2864
5.569std::logical_and< _Tp > Struct Template Reference	2865
5.569.1 Detailed Description	2865
5.569.2 Member Typedef Documentation	2866
5.569.2.1 first_argument_type	2866

5.569.2.2 result_type	2866
5.569.2.3 second_argument_type	2866
5.570std::logical_not< _Tp > Struct Template Reference	2867
5.570.1 Detailed Description	2867
5.570.2 Member Typedef Documentation	2868
5.570.2.1 argument_type	2868
5.570.2.2 result_type	2868
5.571std::logical_or< _Tp > Struct Template Reference	2869
5.571.1 Detailed Description	2869
5.571.2 Member Typedef Documentation	2870
5.571.2.1 first_argument_type	2870
5.571.2.2 result_type	2870
5.571.2.3 second_argument_type	2870
5.572std::lognormal_distribution< _RealType > Class Template Reference	2871
5.572.1 Detailed Description	2872
5.572.2 Member Typedef Documentation	2872
5.572.2.1 result_type	2872
5.572.3 Member Function Documentation	2872
5.572.3.1 max	2872
5.572.3.2 min	2872
5.572.3.3 param	2872
5.572.3.4 param	2873
5.572.3.5 reset	2873
5.572.4 Friends And Related Function Documentation	2873
5.572.4.1 operator<<	2873
5.572.4.2 operator>>	2874
5.573std::lognormal_distribution< _RealType >::param_type Struct Refer- ence	2875
5.573.1 Detailed Description	2875
5.574std::make_signed< _Tp > Struct Template Reference	2876
5.574.1 Detailed Description	2876

5.575	std::make_unsigned< _Tp > Struct Template Reference	2877
5.575.1	Detailed Description	2877
5.576	std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2878
5.576.1	Detailed Description	2880
5.576.2	Constructor & Destructor Documentation	2880
5.576.2.1	map	2880
5.576.2.2	map	2881
5.576.2.3	map	2881
5.576.2.4	map	2881
5.576.2.5	map	2882
5.576.2.6	map	2882
5.576.2.7	map	2882
5.576.3	Member Function Documentation	2883
5.576.3.1	at	2883
5.576.3.2	begin	2883
5.576.3.3	begin	2884
5.576.3.4	cbegin	2884
5.576.3.5	cend	2884
5.576.3.6	clear	2884
5.576.3.7	count	2885
5.576.3.8	crbegin	2885
5.576.3.9	crend	2885
5.576.3.10	empty	2885
5.576.3.11	lend	2886
5.576.3.12	end	2886
5.576.3.13	equal_range	2886
5.576.3.14	equal_range	2887
5.576.3.15	erase	2887
5.576.3.16	erase	2888
5.576.3.17	erase	2888
5.576.3.18	find	2889

5.576.3.19	<code>find</code>	2889
5.576.3.20	<code>get_allocator</code>	2889
5.576.3.2	<code>insert</code>	2890
5.576.3.22	<code>insert</code>	2890
5.576.3.23	<code>insert</code>	2891
5.576.3.24	<code>insert</code>	2891
5.576.3.25	<code>key_comp</code>	2891
5.576.3.26	<code>lower_bound</code>	2892
5.576.3.27	<code>lower_bound</code>	2892
5.576.3.28	<code>max_size</code>	2893
5.576.3.29	<code>operator=</code>	2893
5.576.3.30	<code>operator=</code>	2893
5.576.3.31	<code>operator=</code>	2894
5.576.3.32	<code>operator[]</code>	2894
5.576.3.33	<code>rbegin</code>	2895
5.576.3.34	<code>rbegin</code>	2895
5.576.3.35	<code>rend</code>	2895
5.576.3.36	<code>rend</code>	2895
5.576.3.37	<code>size</code>	2896
5.576.3.38	<code>swap</code>	2896
5.576.3.39	<code>upper_bound</code>	2896
5.576.3.40	<code>upper_bound</code>	2897
5.576.3.41	<code>value_comp</code>	2897
5.577	<code>std::mask_array<_Tp></code> Class Template Reference	2898
5.577.1	Detailed Description	2899
5.578	<code>std::match_results<_Bi_iter, _Allocator></code> Class Template Reference	2900
5.578.1	Detailed Description	2903
5.578.2	Constructor & Destructor Documentation	2904
5.578.2.1	<code>match_results</code>	2904
5.578.2.2	<code>match_results</code>	2904
5.578.2.3	<code>~match_results</code>	2904

5.578.3 Member Function Documentation	2905
5.578.3.1 begin	2905
5.578.3.2 cbegin	2905
5.578.3.3 cend	2905
5.578.3.4 empty	2905
5.578.3.5 end	2906
5.578.3.6 format	2906
5.578.3.7 format	2906
5.578.3.8 length	2906
5.578.3.9 operator=	2907
5.578.3.10operator[]	2907
5.578.3.11position	2907
5.578.3.12prefix	2908
5.578.3.13size	2908
5.578.3.14str	2908
5.578.3.15suffix	2909
5.578.3.16swap	2909
5.579std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2910
5.579.1 Detailed Description	2910
5.579.2 Member Typedef Documentation	2911
5.579.2.1 first_argument_type	2911
5.579.2.2 result_type	2911
5.579.2.3 second_argument_type	2911
5.580std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2912
5.580.1 Detailed Description	2912
5.580.2 Member Typedef Documentation	2913
5.580.2.1 first_argument_type	2913
5.580.2.2 result_type	2913
5.580.2.3 second_argument_type	2913
5.581std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2914
5.581.1 Detailed Description	2914

5.581.2 Member Typedef Documentation	2915
5.581.2.1 argument_type	2915
5.581.2.2 result_type	2915
5.582std::mem_fun_t< _Ret, _Tp > Class Template Reference	2916
5.582.1 Detailed Description	2916
5.582.2 Member Typedef Documentation	2917
5.582.2.1 argument_type	2917
5.582.2.2 result_type	2917
5.583std::messages< _CharT > Class Template Reference	2918
5.583.1 Detailed Description	2920
5.583.2 Member Typedef Documentation	2920
5.583.2.1 char_type	2920
5.583.2.2 string_type	2920
5.583.3 Constructor & Destructor Documentation	2920
5.583.3.1 messages	2920
5.583.3.2 messages	2921
5.583.3.3 ~messages	2921
5.583.4 Member Data Documentation	2921
5.583.4.1 id	2921
5.584std::messages_base Struct Reference	2922
5.584.1 Detailed Description	2922
5.585std::messages_byname< _CharT > Class Template Reference	2923
5.585.1 Detailed Description	2924
5.585.2 Member Typedef Documentation	2925
5.585.2.1 char_type	2925
5.585.2.2 string_type	2925
5.585.3 Member Data Documentation	2925
5.585.3.1 id	2925
5.586std::minus< _Tp > Struct Template Reference	2926
5.586.1 Detailed Description	2926
5.586.2 Member Typedef Documentation	2927

5.586.2.1	first_argument_type	2927
5.586.2.2	result_type	2927
5.586.2.3	second_argument_type	2927
5.587	std::modulus< _Tp > Struct Template Reference	2928
5.587.1	Detailed Description	2928
5.587.2	Member Typedef Documentation	2929
5.587.2.1	first_argument_type	2929
5.587.2.2	result_type	2929
5.587.2.3	second_argument_type	2929
5.588	std::money_base Class Reference	2930
5.588.1	Detailed Description	2931
5.589	std::money_get< _CharT, _InIter > Class Template Reference	2932
5.589.1	Detailed Description	2933
5.589.2	Member Typedef Documentation	2933
5.589.2.1	char_type	2933
5.589.2.2	iter_type	2934
5.589.2.3	string_type	2934
5.589.3	Constructor & Destructor Documentation	2934
5.589.3.1	money_get	2934
5.589.3.2	~money_get	2934
5.589.4	Member Function Documentation	2935
5.589.4.1	do_get	2935
5.589.4.2	do_get	2935
5.589.4.3	get	2935
5.589.4.4	get	2936
5.589.5	Member Data Documentation	2937
5.589.5.1	id	2937
5.590	std::money_put< _CharT, _OutIter > Class Template Reference	2938
5.590.1	Detailed Description	2939
5.590.2	Member Typedef Documentation	2939
5.590.2.1	char_type	2939

5.590.2.2	iter_type	2940
5.590.2.3	string_type	2940
5.590.3	Constructor & Destructor Documentation	2940
5.590.3.1	money_put	2940
5.590.3.2	~money_put	2940
5.590.4	Member Function Documentation	2941
5.590.4.1	do_put	2941
5.590.4.2	do_put	2941
5.590.4.3	put	2942
5.590.4.4	put	2943
5.590.5	Member Data Documentation	2943
5.590.5.1	id	2943
5.591	std::moneypunct< _CharT, _Intl > Class Template Reference	2944
5.591.1	Detailed Description	2946
5.591.2	Member Typedef Documentation	2946
5.591.2.1	char_type	2946
5.591.2.2	string_type	2947
5.591.3	Constructor & Destructor Documentation	2947
5.591.3.1	moneypunct	2947
5.591.3.2	moneypunct	2947
5.591.3.3	moneypunct	2947
5.591.3.4	~moneypunct	2948
5.591.4	Member Function Documentation	2948
5.591.4.1	curr_symbol	2948
5.591.4.2	decimal_point	2948
5.591.4.3	do_curr_symbol	2949
5.591.4.4	do_decimal_point	2949
5.591.4.5	do_frac_digits	2949
5.591.4.6	do_grouping	2950
5.591.4.7	do_neg_format	2950
5.591.4.8	do_negative_sign	2951

5.591.4.9 do_pos_format	2951
5.591.4.10do_positive_sign	2951
5.591.4.11do_thousands_sep	2952
5.591.4.12frac_digits	2952
5.591.4.13grouping	2953
5.591.4.14neg_format	2953
5.591.4.15negative_sign	2953
5.591.4.16pos_format	2954
5.591.4.17positive_sign	2954
5.591.4.18thousands_sep	2955
5.591.5 Member Data Documentation	2955
5.591.5.1 id	2955
5.591.5.2 intl	2955
5.592std::money_punct_byname< _CharT, _Intl > Class Template Reference	2957
5.592.1 Detailed Description	2959
5.592.2 Member Typedef Documentation	2959
5.592.2.1 char_type	2959
5.592.2.2 string_type	2959
5.592.3 Member Function Documentation	2960
5.592.3.1 curr_symbol	2960
5.592.3.2 decimal_point	2960
5.592.3.3 do_curr_symbol	2960
5.592.3.4 do_decimal_point	2961
5.592.3.5 do_frac_digits	2961
5.592.3.6 do_grouping	2961
5.592.3.7 do_neg_format	2962
5.592.3.8 do_negative_sign	2962
5.592.3.9 do_pos_format	2963
5.592.3.10do_positive_sign	2963
5.592.3.11do_thousands_sep	2963
5.592.3.12frac_digits	2964

5.592.3.13	grouping	2964
5.592.3.14	neg_format	2965
5.592.3.15	negative_sign	2965
5.592.3.16	pos_format	2965
5.592.3.17	positive_sign	2966
5.592.3.18	thousands_sep	2966
5.592.4	Member Data Documentation	2967
5.592.4.1	id	2967
5.592.4.2	intl	2967
5.593	std::move_iterator< _Iterator > Class Template Reference	2968
5.593.1	Detailed Description	2969
5.594	std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2970
5.594.1	Detailed Description	2972
5.594.2	Constructor & Destructor Documentation	2972
5.594.2.1	multimap	2972
5.594.2.2	multimap	2973
5.594.2.3	multimap	2973
5.594.2.4	multimap	2973
5.594.2.5	multimap	2974
5.594.2.6	multimap	2974
5.594.2.7	multimap	2974
5.594.3	Member Function Documentation	2975
5.594.3.1	begin	2975
5.594.3.2	begin	2975
5.594.3.3	cbegin	2975
5.594.3.4	cend	2976
5.594.3.5	clear	2976
5.594.3.6	count	2976
5.594.3.7	crbegin	2976
5.594.3.8	crend	2977

5.594.3.9	empty	2977
5.594.3.10	end	2977
5.594.3.11	lend	2977
5.594.3.12	equal_range	2978
5.594.3.13	equal_range	2978
5.594.3.14	erase	2979
5.594.3.15	erase	2979
5.594.3.16	erase	2980
5.594.3.17	find	2980
5.594.3.18	find	2981
5.594.3.19	get_allocator	2981
5.594.3.20	insert	2981
5.594.3.21	insert	2982
5.594.3.22	insert	2982
5.594.3.23	insert	2983
5.594.3.24	key_comp	2983
5.594.3.25	lower_bound	2983
5.594.3.26	lower_bound	2984
5.594.3.27	max_size	2984
5.594.3.28	operator=	2984
5.594.3.29	operator=	2985
5.594.3.30	operator=	2985
5.594.3.31	lrbegin	2986
5.594.3.32	rbegin	2986
5.594.3.33	rend	2986
5.594.3.34	rend	2986
5.594.3.35	size	2987
5.594.3.36	swap	2987
5.594.3.37	upper_bound	2987
5.594.3.38	upper_bound	2988
5.594.3.39	value_comp	2988

5.595std::multiplies<_Tp> Struct Template Reference	2989
5.595.1 Detailed Description	2989
5.595.2 Member Typedef Documentation	2990
5.595.2.1 first_argument_type	2990
5.595.2.2 result_type	2990
5.595.2.3 second_argument_type	2990
5.596std::multiset<_Key, _Compare, _Alloc> Class Template Reference .	2991
5.596.1 Detailed Description	2993
5.596.2 Constructor & Destructor Documentation	2993
5.596.2.1 multiset	2993
5.596.2.2 multiset	2993
5.596.2.3 multiset	2994
5.596.2.4 multiset	2994
5.596.2.5 multiset	2994
5.596.2.6 multiset	2995
5.596.2.7 multiset	2995
5.596.3 Member Function Documentation	2996
5.596.3.1 begin	2996
5.596.3.2 cbegin	2996
5.596.3.3 cend	2996
5.596.3.4 clear	2996
5.596.3.5 count	2996
5.596.3.6 crbegin	2997
5.596.3.7 crend	2997
5.596.3.8 empty	2997
5.596.3.9 end	2997
5.596.3.10equal_range	2998
5.596.3.11lequal_range	2998
5.596.3.12erase	2999
5.596.3.13erase	2999
5.596.3.14erase	3000

5.596.3.15	find	3000
5.596.3.16	find	3001
5.596.3.17	get_allocator	3001
5.596.3.18	insert	3001
5.596.3.19	insert	3002
5.596.3.20	insert	3002
5.596.3.21	insert	3003
5.596.3.22	key_comp	3003
5.596.3.23	lower_bound	3003
5.596.3.24	lower_bound	3004
5.596.3.25	max_size	3004
5.596.3.26	operator=	3004
5.596.3.27	operator=	3005
5.596.3.28	operator=	3005
5.596.3.29	begin	3005
5.596.3.30	end	3006
5.596.3.31	size	3006
5.596.3.32	swap	3006
5.596.3.33	upper_bound	3006
5.596.3.34	upper_bound	3007
5.596.3.35	value_comp	3007
5.596.4	Friends And Related Function Documentation	3008
5.596.4.1	operator<	3008
5.596.4.2	operator==	3008
5.597	std::mutex Class Reference	3010
5.597.1	Detailed Description	3010
5.598	std::negate< _Tp > Struct Template Reference	3011
5.598.1	Detailed Description	3011
5.598.2	Member Typedef Documentation	3012
5.598.2.1	argument_type	3012
5.598.2.2	result_type	3012

5.599std::negative_binomial_distribution< _IntType > Class Template Reference	3013
5.599.1 Detailed Description	3014
5.599.2 Member Typedef Documentation	3014
5.599.2.1 result_type	3014
5.599.3 Member Function Documentation	3014
5.599.3.1 k	3014
5.599.3.2 max	3014
5.599.3.3 min	3014
5.599.3.4 p	3015
5.599.3.5 param	3015
5.599.3.6 param	3015
5.599.3.7 reset	3015
5.599.4 Friends And Related Function Documentation	3016
5.599.4.1 operator<<	3016
5.599.4.2 operator>>	3016
5.600std::negative_binomial_distribution< _IntType >::param_type Struct Reference	3017
5.600.1 Detailed Description	3017
5.601std::nested_exception Class Reference	3018
5.601.1 Detailed Description	3018
5.602std::normal_distribution< _RealType > Class Template Reference	3019
5.602.1 Detailed Description	3020
5.602.2 Member Typedef Documentation	3020
5.602.2.1 result_type	3020
5.602.3 Constructor & Destructor Documentation	3020
5.602.3.1 normal_distribution	3020
5.602.4 Member Function Documentation	3021
5.602.4.1 max	3021
5.602.4.2 mean	3021
5.602.4.3 min	3021
5.602.4.4 operator()	3021

5.602.4.5 param	3021
5.602.4.6 param	3022
5.602.4.7 reset	3022
5.602.4.8 stddev	3022
5.602.5 Friends And Related Function Documentation	3022
5.602.5.1 operator<<	3022
5.602.5.2 operator>>	3023
5.603std::normal_distribution< _RealType >::param_type Struct Reference	3024
5.603.1 Detailed Description	3024
5.604std::not_equal_to< _Tp > Struct Template Reference	3025
5.604.1 Detailed Description	3025
5.604.2 Member Typedef Documentation	3026
5.604.2.1 first_argument_type	3026
5.604.2.2 result_type	3026
5.604.2.3 second_argument_type	3026
5.605std::num_get< _CharT, _InIter > Class Template Reference	3027
5.605.1 Detailed Description	3029
5.605.2 Member Typedef Documentation	3029
5.605.2.1 char_type	3029
5.605.2.2 iter_type	3030
5.605.3 Constructor & Destructor Documentation	3030
5.605.3.1 num_get	3030
5.605.3.2 ~num_get	3030
5.605.4 Member Function Documentation	3030
5.605.4.1 do_get	3030
5.605.4.2 do_get	3031
5.605.4.3 do_get	3031
5.605.4.4 do_get	3031
5.605.4.5 do_get	3031
5.605.4.6 do_get	3032
5.605.4.7 do_get	3032

5.605.4.8 do_get	3032
5.605.4.9 do_get	3032
5.605.4.10do_get	3032
5.605.4.11do_get	3033
5.605.4.12get	3033
5.605.4.13get	3034
5.605.4.14get	3034
5.605.4.15get	3034
5.605.4.16get	3035
5.605.4.17get	3035
5.605.4.18get	3036
5.605.4.19get	3036
5.605.4.20get	3036
5.605.4.21get	3036
5.605.4.22get	3037
5.605.5 Member Data Documentation	3038
5.605.5.1 id	3038
5.606std::num_put< _CharT, _OutIter > Class Template Reference	3039
5.606.1 Detailed Description	3041
5.606.2 Member Typedef Documentation	3041
5.606.2.1 char_type	3041
5.606.2.2 iter_type	3041
5.606.3 Constructor & Destructor Documentation	3041
5.606.3.1 num_put	3041
5.606.3.2 ~num_put	3042
5.606.4 Member Function Documentation	3042
5.606.4.1 do_put	3042
5.606.4.2 do_put	3042
5.606.4.3 do_put	3042
5.606.4.4 do_put	3043
5.606.4.5 do_put	3043

5.606.4.6 do_put	3043
5.606.4.7 do_put	3043
5.606.4.8 do_put	3043
5.606.4.9 put	3044
5.606.4.10put	3044
5.606.4.11put	3045
5.606.4.12put	3046
5.606.4.13put	3046
5.606.4.14put	3046
5.606.4.15put	3046
5.606.4.16put	3047
5.606.5 Member Data Documentation	3048
5.606.5.1 id	3048
5.607std::numeric_limits< _Tp > Struct Template Reference	3049
5.607.1 Detailed Description	3050
5.607.2 Member Function Documentation	3050
5.607.2.1 denorm_min	3050
5.607.2.2 epsilon	3050
5.607.2.3 infinity	3051
5.607.2.4 lowest	3051
5.607.2.5 max	3051
5.607.2.6 min	3051
5.607.2.7 quiet_NaN	3051
5.607.2.8 round_error	3051
5.607.2.9 signaling_NaN	3052
5.607.3 Member Data Documentation	3052
5.607.3.1 digits	3052
5.607.3.2 digits10	3052
5.607.3.3 has_denorm	3052
5.607.3.4 has_denorm_loss	3052
5.607.3.5 has_infinity	3052

5.607.3.6	has_quiet_NaN	3053
5.607.3.7	has_signaling_NaN	3053
5.607.3.8	is_bounded	3053
5.607.3.9	is_exact	3053
5.607.3.10	is_iec559	3053
5.607.3.11	is_integer	3054
5.607.3.12	is_modulo	3054
5.607.3.13	is_signed	3054
5.607.3.14	is_specialized	3054
5.607.3.15	max_digits10	3054
5.607.3.16	max_exponent	3054
5.607.3.17	max_exponent10	3055
5.607.3.18	min_exponent	3055
5.607.3.19	min_exponent10	3055
5.607.3.20	radix	3055
5.607.3.21	round_style	3055
5.607.3.22	traps_before	3056
5.607.3.23	traps	3056
5.608	std::numeric_limits< bool > Struct Template Reference	3057
5.608.1	Detailed Description	3058
5.609	std::numeric_limits< char > Struct Template Reference	3059
5.609.1	Detailed Description	3060
5.610	std::numeric_limits< char16_t > Struct Template Reference	3061
5.610.1	Detailed Description	3062
5.611	std::numeric_limits< char32_t > Struct Template Reference	3063
5.611.1	Detailed Description	3064
5.612	std::numeric_limits< double > Struct Template Reference	3065
5.612.1	Detailed Description	3066
5.613	std::numeric_limits< float > Struct Template Reference	3067
5.613.1	Detailed Description	3068
5.614	std::numeric_limits< int > Struct Template Reference	3069

5.614.1 Detailed Description	3070
5.615std::numeric_limits< long > Struct Template Reference	3071
5.615.1 Detailed Description	3072
5.616std::numeric_limits< long double > Struct Template Reference	3073
5.616.1 Detailed Description	3074
5.617std::numeric_limits< long long > Struct Template Reference	3075
5.617.1 Detailed Description	3076
5.618std::numeric_limits< short > Struct Template Reference	3077
5.618.1 Detailed Description	3078
5.619std::numeric_limits< signed char > Struct Template Reference	3079
5.619.1 Detailed Description	3080
5.620std::numeric_limits< unsigned char > Struct Template Reference	3081
5.620.1 Detailed Description	3082
5.621std::numeric_limits< unsigned int > Struct Template Reference	3083
5.621.1 Detailed Description	3084
5.622std::numeric_limits< unsigned long > Struct Template Reference	3085
5.622.1 Detailed Description	3086
5.623std::numeric_limits< unsigned long long > Struct Template Reference	3087
5.623.1 Detailed Description	3088
5.624std::numeric_limits< unsigned short > Struct Template Reference	3089
5.624.1 Detailed Description	3090
5.625std::numeric_limits< wchar_t > Struct Template Reference	3091
5.625.1 Detailed Description	3092
5.626std::num_punct< _CharT > Class Template Reference	3093
5.626.1 Detailed Description	3095
5.626.2 Member Typedef Documentation	3095
5.626.2.1 char_type	3095
5.626.2.2 string_type	3095
5.626.3 Constructor & Destructor Documentation	3095
5.626.3.1 num_punct	3095
5.626.3.2 num_punct	3096

5.626.3.3	numpunct	3096
5.626.3.4	~numpunct	3096
5.626.4	Member Function Documentation	3096
5.626.4.1	decimal_point	3096
5.626.4.2	do_decimal_point	3097
5.626.4.3	do_falsename	3097
5.626.4.4	do_grouping	3097
5.626.4.5	do_thousands_sep	3098
5.626.4.6	do_truename	3098
5.626.4.7	falsename	3098
5.626.4.8	grouping	3099
5.626.4.9	thousands_sep	3099
5.626.4.10	truename	3100
5.626.5	Member Data Documentation	3100
5.626.5.1	id	3100
5.627	std::numpunct_byname<_CharT> Class Template Reference	3101
5.627.1	Detailed Description	3102
5.627.2	Member Typedef Documentation	3103
5.627.2.1	char_type	3103
5.627.2.2	string_type	3103
5.627.3	Member Function Documentation	3103
5.627.3.1	decimal_point	3103
5.627.3.2	do_decimal_point	3103
5.627.3.3	do_falsename	3104
5.627.3.4	do_grouping	3104
5.627.3.5	do_thousands_sep	3104
5.627.3.6	do_truename	3105
5.627.3.7	falsename	3105
5.627.3.8	grouping	3105
5.627.3.9	thousands_sep	3106
5.627.3.10	truename	3106

5.627.4 Member Data Documentation	3107
5.627.4.1 id	3107
5.628std::once_flag Struct Reference	3108
5.628.1 Detailed Description	3108
5.629std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	3109
5.629.1 Detailed Description	3110
5.629.2 Member Typedef Documentation	3110
5.629.2.1 char_type	3110
5.629.2.2 difference_type	3110
5.629.2.3 iterator_category	3110
5.629.2.4 ostream_type	3111
5.629.2.5 pointer	3111
5.629.2.6 reference	3111
5.629.2.7 traits_type	3111
5.629.2.8 value_type	3111
5.629.3 Constructor & Destructor Documentation	3112
5.629.3.1 ostream_iterator	3112
5.629.3.2 ostream_iterator	3112
5.629.3.3 ostream_iterator	3112
5.629.4 Member Function Documentation	3112
5.629.4.1 operator=	3112
5.630std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	3114
5.630.1 Detailed Description	3115
5.630.2 Member Typedef Documentation	3115
5.630.2.1 char_type	3115
5.630.2.2 difference_type	3115
5.630.2.3 iterator_category	3115
5.630.2.4 ostream_type	3115
5.630.2.5 pointer	3116
5.630.2.6 reference	3116
5.630.2.7 streambuf_type	3116

5.630.2.8 traits_type	3116
5.630.2.9 value_type	3116
5.630.3 Constructor & Destructor Documentation	3117
5.630.3.1 ostreambuf_iterator	3117
5.630.3.2 ostreambuf_iterator	3117
5.630.4 Member Function Documentation	3117
5.630.4.1 failed	3117
5.630.4.2 operator*	3117
5.630.4.3 operator++	3117
5.630.4.4 operator++	3118
5.630.4.5 operator=	3118
5.631 std::out_of_range Class Reference	3119
5.631.1 Detailed Description	3119
5.631.2 Member Function Documentation	3119
5.631.2.1 what	3119
5.632 std::output_iterator_tag Struct Reference	3121
5.632.1 Detailed Description	3121
5.633 std::overflow_error Class Reference	3122
5.633.1 Detailed Description	3122
5.633.2 Member Function Documentation	3122
5.633.2.1 what	3122
5.634 std::owner_less< shared_ptr< _Tp > > Struct Template Reference	3124
5.634.1 Detailed Description	3124
5.635 std::owner_less< weak_ptr< _Tp > > Struct Template Reference	3125
5.635.1 Detailed Description	3125
5.636 std::packaged_task< _Res(_ArgTypes...) > Class Template Reference	3126
5.636.1 Detailed Description	3126
5.637 std::pair< _T1, _T2 > Struct Template Reference	3127
5.637.1 Detailed Description	3128
5.637.2 Member Typedef Documentation	3128
5.637.2.1 first_type	3128

5.637.2.2	second_type	3128
5.637.3	Constructor & Destructor Documentation	3128
5.637.3.1	pair	3128
5.637.3.2	pair	3128
5.637.3.3	pair	3129
5.637.4	Member Data Documentation	3129
5.637.4.1	first	3129
5.637.4.2	second	3129
5.638	std::piecewise_constant_distribution< _RealType > Class Template Reference	3130
5.638.1	Detailed Description	3131
5.638.2	Member Typedef Documentation	3131
5.638.2.1	result_type	3131
5.638.3	Member Function Documentation	3131
5.638.3.1	densities	3131
5.638.3.2	intervals	3131
5.638.3.3	max	3132
5.638.3.4	min	3132
5.638.3.5	param	3132
5.638.3.6	param	3132
5.638.3.7	reset	3133
5.638.4	Friends And Related Function Documentation	3133
5.638.4.1	operator<<	3133
5.638.4.2	operator>>	3133
5.639	std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	3135
5.639.1	Detailed Description	3135
5.640	std::piecewise_linear_distribution< _RealType > Class Template Reference	3136
5.640.1	Detailed Description	3137
5.640.2	Member Typedef Documentation	3137
5.640.2.1	result_type	3137

5.640.3 Member Function Documentation	3137
5.640.3.1 densities	3137
5.640.3.2 intervals	3137
5.640.3.3 max	3138
5.640.3.4 min	3138
5.640.3.5 param	3138
5.640.3.6 param	3138
5.640.3.7 reset	3139
5.640.4 Friends And Related Function Documentation	3139
5.640.4.1 operator<<	3139
5.640.4.2 operator>>	3139
5.641std::piecewise_linear_distribution< _RealType >::param_type Struct Reference	3141
5.641.1 Detailed Description	3141
5.642std::plus< _Tp > Struct Template Reference	3142
5.642.1 Detailed Description	3142
5.642.2 Member Typedef Documentation	3143
5.642.2.1 first_argument_type	3143
5.642.2.2 result_type	3143
5.642.2.3 second_argument_type	3143
5.643std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	3144
5.643.1 Detailed Description	3144
5.643.2 Member Typedef Documentation	3145
5.643.2.1 first_argument_type	3145
5.643.2.2 result_type	3145
5.643.2.3 second_argument_type	3145
5.644std::pointer_to_unary_function< _Arg, _Result > Class Template Ref- erence	3146
5.644.1 Detailed Description	3146
5.644.2 Member Typedef Documentation	3147
5.644.2.1 argument_type	3147

5.644.2.2 result_type	3147
5.645std::poisson_distribution< _IntType > Class Template Reference . . .	3148
5.645.1 Detailed Description	3149
5.645.2 Member Typedef Documentation	3149
5.645.2.1 result_type	3149
5.645.3 Member Function Documentation	3149
5.645.3.1 max	3149
5.645.3.2 mean	3149
5.645.3.3 min	3149
5.645.3.4 operator()	3150
5.645.3.5 param	3150
5.645.3.6 param	3150
5.645.3.7 reset	3150
5.645.4 Friends And Related Function Documentation	3151
5.645.4.1 operator<<	3151
5.645.4.2 operator>>	3151
5.646std::poisson_distribution< _IntType >::param_type Struct Reference	3152
5.646.1 Detailed Description	3152
5.647std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	3153
5.647.1 Detailed Description	3154
5.647.2 Constructor & Destructor Documentation	3154
5.647.2.1 priority_queue	3154
5.647.2.2 priority_queue	3155
5.647.3 Member Function Documentation	3155
5.647.3.1 empty	3155
5.647.3.2 pop	3155
5.647.3.3 push	3156
5.647.3.4 size	3156
5.647.3.5 top	3156
5.648std::promise< _Res > Class Template Reference	3158

5.648.1 Detailed Description	3158
5.649std::promise< _Res & > Class Template Reference	3159
5.649.1 Detailed Description	3159
5.650std::promise< void > Class Template Reference	3160
5.650.1 Detailed Description	3160
5.651std::queue< _Tp, _Sequence > Class Template Reference	3161
5.651.1 Detailed Description	3162
5.651.2 Constructor & Destructor Documentation	3162
5.651.2.1 queue	3162
5.651.3 Member Function Documentation	3163
5.651.3.1 back	3163
5.651.3.2 back	3163
5.651.3.3 empty	3163
5.651.3.4 front	3163
5.651.3.5 front	3163
5.651.3.6 pop	3163
5.651.3.7 push	3164
5.651.3.8 size	3164
5.651.4 Member Data Documentation	3164
5.651.4.1 c	3164
5.652std::random_access_iterator_tag Struct Reference	3165
5.652.1 Detailed Description	3165
5.653std::random_device Class Reference	3166
5.653.1 Detailed Description	3166
5.653.2 Member Typedef Documentation	3166
5.653.2.1 result_type	3166
5.654std::range_error Class Reference	3167
5.654.1 Detailed Description	3167
5.654.2 Member Function Documentation	3167
5.654.2.1 what	3167
5.655std::rank< typename > Struct Template Reference	3169

5.655.1 Detailed Description	3169
5.656std::ratio< _Num, _Den > Struct Template Reference	3170
5.656.1 Detailed Description	3170
5.657std::ratio_add< _R1, _R2 > Struct Template Reference	3171
5.657.1 Detailed Description	3171
5.658std::ratio_divide< _R1, _R2 > Struct Template Reference	3172
5.658.1 Detailed Description	3172
5.659std::ratio_equal< _R1, _R2 > Struct Template Reference	3173
5.659.1 Detailed Description	3173
5.660std::ratio_greater< _R1, _R2 > Struct Template Reference	3174
5.660.1 Detailed Description	3174
5.661std::ratio_greater_equal< _R1, _R2 > Struct Template Reference	3175
5.661.1 Detailed Description	3175
5.662std::ratio_less< _R1, _R2 > Struct Template Reference	3176
5.662.1 Detailed Description	3176
5.663std::ratio_less_equal< _R1, _R2 > Struct Template Reference	3177
5.663.1 Detailed Description	3177
5.664std::ratio_multiply< _R1, _R2 > Struct Template Reference	3178
5.664.1 Detailed Description	3178
5.665std::ratio_not_equal< _R1, _R2 > Struct Template Reference	3179
5.665.1 Detailed Description	3179
5.666std::ratio_subtract< _R1, _R2 > Struct Template Reference	3180
5.666.1 Detailed Description	3180
5.667std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	3181
5.667.1 Detailed Description	3182
5.667.2 Member Typedef Documentation	3182
5.667.2.1 difference_type	3182
5.667.2.2 iterator_category	3182
5.667.2.3 pointer	3182
5.667.2.4 reference	3182

5.667.2.5 value_type	3182
5.668std::recursive_mutex Class Reference	3184
5.668.1 Detailed Description	3184
5.669std::recursive_timed_mutex Class Reference	3185
5.669.1 Detailed Description	3185
5.670std::reference_wrapper< _Tp > Class Template Reference	3186
5.670.1 Detailed Description	3187
5.671std::regex_error Class Reference	3188
5.671.1 Detailed Description	3188
5.671.2 Constructor & Destructor Documentation	3189
5.671.2.1 regex_error	3189
5.671.3 Member Function Documentation	3189
5.671.3.1 code	3189
5.671.3.2 what	3189
5.672std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3190
5.672.1 Detailed Description	3190
5.672.2 Constructor & Destructor Documentation	3191
5.672.2.1 regex_iterator	3191
5.672.2.2 regex_iterator	3191
5.672.2.3 regex_iterator	3191
5.672.3 Member Function Documentation	3192
5.672.3.1 operator!=	3192
5.672.3.2 operator*	3192
5.672.3.3 operator++	3192
5.672.3.4 operator++	3193
5.672.3.5 operator->	3193
5.672.3.6 operator=	3193
5.672.3.7 operator==	3193
5.673std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3195
5.673.1 Detailed Description	3195

5.673.2 Constructor & Destructor Documentation	3196
5.673.2.1 regex_token_iterator	3196
5.673.2.2 regex_token_iterator	3196
5.673.2.3 regex_token_iterator	3197
5.673.2.4 regex_token_iterator	3197
5.673.2.5 regex_token_iterator	3198
5.673.3 Member Function Documentation	3198
5.673.3.1 operator!=	3198
5.673.3.2 operator*	3199
5.673.3.3 operator++	3199
5.673.3.4 operator++	3199
5.673.3.5 operator->	3199
5.673.3.6 operator=	3200
5.673.3.7 operator==	3200
5.674std::regex_traits< _Ch_type > Struct Template Reference	3201
5.674.1 Detailed Description	3202
5.674.2 Constructor & Destructor Documentation	3202
5.674.2.1 regex_traits	3202
5.674.3 Member Function Documentation	3202
5.674.3.1 getloc	3202
5.674.3.2 imbue	3202
5.674.3.3 length	3203
5.674.3.4 lookup_classname	3203
5.674.3.5 lookup_collatename	3204
5.674.3.6 transform	3205
5.674.3.7 transform_primary	3205
5.674.3.8 translate	3206
5.674.3.9 translate_nocase	3206
5.675std::remove_all_extents< _Tp > Struct Template Reference	3207
5.675.1 Detailed Description	3207
5.676std::remove_const< _Tp > Struct Template Reference	3208

5.676.1 Detailed Description	3208
5.677std::remove_cv< _Tp > Struct Template Reference	3209
5.677.1 Detailed Description	3209
5.678std::remove_extent< _Tp > Struct Template Reference	3210
5.678.1 Detailed Description	3210
5.679std::remove_pointer< _Tp > Struct Template Reference	3211
5.679.1 Detailed Description	3211
5.680std::remove_reference< _Tp > Struct Template Reference	3212
5.680.1 Detailed Description	3212
5.681std::remove_volatile< _Tp > Struct Template Reference	3213
5.681.1 Detailed Description	3213
5.682std::reverse_iterator< _Iterator > Class Template Reference	3214
5.682.1 Detailed Description	3215
5.682.2 Member Typedef Documentation	3215
5.682.2.1 difference_type	3215
5.682.2.2 iterator_category	3215
5.682.2.3 pointer	3216
5.682.2.4 reference	3216
5.682.2.5 value_type	3216
5.682.3 Constructor & Destructor Documentation	3216
5.682.3.1 reverse_iterator	3216
5.682.3.2 reverse_iterator	3217
5.682.3.3 reverse_iterator	3217
5.682.3.4 reverse_iterator	3217
5.682.4 Member Function Documentation	3217
5.682.4.1 base	3217
5.682.4.2 operator*	3217
5.682.4.3 operator+	3218
5.682.4.4 operator++	3218
5.682.4.5 operator++	3218
5.682.4.6 operator+=	3219

5.682.4.7 operator-	3219
5.682.4.8 operator--	3219
5.682.4.9 operator--	3220
5.682.4.10operator==	3220
5.682.4.11operator->	3220
5.682.4.12operator[]	3220
5.683std::runtime_error Class Reference	3222
5.683.1 Detailed Description	3222
5.683.2 Constructor & Destructor Documentation	3222
5.683.2.1 runtime_error	3222
5.683.3 Member Function Documentation	3223
5.683.3.1 what	3223
5.684std::seed_seq Class Reference	3224
5.684.1 Detailed Description	3224
5.684.2 Member Typedef Documentation	3224
5.684.2.1 result_type	3224
5.684.3 Constructor & Destructor Documentation	3224
5.684.3.1 seed_seq	3224
5.685std::set<_Key, _Compare, _Alloc > Class Template Reference	3226
5.685.1 Detailed Description	3228
5.685.2 Member Typedef Documentation	3228
5.685.2.1 allocator_type	3228
5.685.2.2 const_iterator	3228
5.685.2.3 const_pointer	3229
5.685.2.4 const_reference	3229
5.685.2.5 const_reverse_iterator	3229
5.685.2.6 difference_type	3229
5.685.2.7 iterator	3229
5.685.2.8 key_compare	3230
5.685.2.9 key_type	3230
5.685.2.10pointer	3230

5.685.2.1	reference	3230
5.685.2.12	reverse_iterator	3230
5.685.2.13	size_type	3231
5.685.2.14	value_compare	3231
5.685.2.15	value_type	3231
5.685.3	Constructor & Destructor Documentation	3231
5.685.3.1	set	3231
5.685.3.2	set	3231
5.685.3.3	set	3232
5.685.3.4	set	3232
5.685.3.5	set	3233
5.685.3.6	set	3233
5.685.3.7	set	3233
5.685.4	Member Function Documentation	3234
5.685.4.1	begin	3234
5.685.4.2	cbegin	3234
5.685.4.3	cend	3234
5.685.4.4	clear	3234
5.685.4.5	count	3234
5.685.4.6	crbegin	3235
5.685.4.7	crend	3235
5.685.4.8	empty	3235
5.685.4.9	end	3235
5.685.4.10	equal_range	3236
5.685.4.11	lequal_range	3236
5.685.4.12	erase	3236
5.685.4.13	erase	3237
5.685.4.14	erase	3237
5.685.4.15	find	3238
5.685.4.16	find	3238
5.685.4.17	get_allocator	3238

5.685.4.18	insert	3239
5.685.4.19	insert	3239
5.685.4.20	insert	3239
5.685.4.21	insert	3240
5.685.4.22	key_comp	3240
5.685.4.23	lower_bound	3241
5.685.4.24	lower_bound	3241
5.685.4.25	max_size	3241
5.685.4.26	operator=	3241
5.685.4.27	operator=	3242
5.685.4.28	operator=	3242
5.685.4.29	begin	3243
5.685.4.30	rend	3243
5.685.4.31	size	3243
5.685.4.32	swap	3243
5.685.4.33	upper_bound	3244
5.685.4.34	upper_bound	3244
5.685.4.35	value_comp	3244
5.686	std::shared_future< _Res > Class Template Reference	3245
5.686.1	Detailed Description	3246
5.686.2	Constructor & Destructor Documentation	3246
5.686.2.1	shared_future	3246
5.686.2.2	shared_future	3246
5.686.2.3	shared_future	3246
5.686.3	Member Function Documentation	3246
5.686.3.1	_M_get_result	3246
5.686.3.2	get	3247
5.687	std::shared_future< _Res & > Class Template Reference	3248
5.687.1	Detailed Description	3249
5.687.2	Constructor & Destructor Documentation	3249
5.687.2.1	shared_future	3249

5.687.2.2	shared_future	3249
5.687.2.3	shared_future	3249
5.687.3	Member Function Documentation	3250
5.687.3.1	_M_get_result	3250
5.687.3.2	get	3250
5.688	std::shared_future< void > Class Template Reference	3251
5.688.1	Detailed Description	3252
5.688.2	Constructor & Destructor Documentation	3252
5.688.2.1	shared_future	3252
5.688.2.2	shared_future	3252
5.688.2.3	shared_future	3252
5.688.3	Member Function Documentation	3253
5.688.3.1	_M_get_result	3253
5.689	std::shared_ptr< _Tp > Class Template Reference	3254
5.689.1	Detailed Description	3255
5.689.2	Constructor & Destructor Documentation	3255
5.689.2.1	shared_ptr	3255
5.689.2.2	shared_ptr	3255
5.689.2.3	shared_ptr	3256
5.689.2.4	shared_ptr	3256
5.689.2.5	shared_ptr	3257
5.689.2.6	shared_ptr	3257
5.689.2.7	shared_ptr	3258
5.689.2.8	shared_ptr	3258
5.689.2.9	shared_ptr	3258
5.689.3	Friends And Related Function Documentation	3259
5.689.3.1	allocate_shared	3259
5.690	std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	3260
5.690.1	Detailed Description	3261
5.690.2	Member Typedef Documentation	3261

5.690.2.1 result_type	3261
5.690.3 Constructor & Destructor Documentation	3261
5.690.3.1 shuffle_order_engine	3261
5.690.3.2 shuffle_order_engine	3262
5.690.3.3 shuffle_order_engine	3262
5.690.3.4 shuffle_order_engine	3262
5.690.3.5 shuffle_order_engine	3263
5.690.4 Member Function Documentation	3263
5.690.4.1 base	3263
5.690.4.2 discard	3263
5.690.4.3 max	3263
5.690.4.4 min	3264
5.690.4.5 operator()	3264
5.690.4.6 seed	3264
5.690.4.7 seed	3264
5.690.4.8 seed	3265
5.690.5 Friends And Related Function Documentation	3265
5.690.5.1 operator<<	3265
5.690.5.2 operator==	3265
5.690.5.3 operator>>	3266
5.691std::slice Class Reference	3267
5.691.1 Detailed Description	3267
5.692std::slice_array< _Tp > Class Template Reference	3268
5.692.1 Detailed Description	3269
5.693std::stack< _Tp, _Sequence > Class Template Reference	3270
5.693.1 Detailed Description	3271
5.693.2 Constructor & Destructor Documentation	3271
5.693.2.1 stack	3271
5.693.3 Member Function Documentation	3271
5.693.3.1 empty	3271
5.693.3.2 pop	3272

5.693.3.3 push	3272
5.693.3.4 size	3272
5.693.3.5 top	3272
5.693.3.6 top	3272
5.694std::student_t_distribution<_RealType > Class Template Reference .	3274
5.694.1 Detailed Description	3275
5.694.2 Member Typedef Documentation	3275
5.694.2.1 result_type	3275
5.694.3 Member Function Documentation	3275
5.694.3.1 max	3275
5.694.3.2 min	3275
5.694.3.3 param	3275
5.694.3.4 param	3276
5.694.3.5 reset	3276
5.694.4 Friends And Related Function Documentation	3276
5.694.4.1 operator<<	3276
5.694.4.2 operator>>	3277
5.695std::student_t_distribution<_RealType >::param_type Struct Reference	3278
5.695.1 Detailed Description	3278
5.696std::sub_match<_BiIter > Class Template Reference	3279
5.696.1 Detailed Description	3280
5.696.2 Member Typedef Documentation	3280
5.696.2.1 first_type	3280
5.696.2.2 second_type	3280
5.696.3 Member Function Documentation	3281
5.696.3.1 compare	3281
5.696.3.2 compare	3281
5.696.3.3 compare	3281
5.696.3.4 length	3282
5.696.3.5 operator basic_string<value_type >	3282
5.696.3.6 str	3282

5.696.4 Member Data Documentation	3283
5.696.4.1 first	3283
5.696.4.2 second	3283
5.697std::system_error Class Reference	3284
5.697.1 Detailed Description	3284
5.697.2 Member Function Documentation	3285
5.697.2.1 what	3285
5.698std::thread Class Reference	3286
5.698.1 Detailed Description	3286
5.698.2 Member Function Documentation	3287
5.698.2.1 native_handle	3287
5.699std::thread::id Class Reference	3288
5.699.1 Detailed Description	3288
5.700std::time_base Class Reference	3289
5.700.1 Detailed Description	3289
5.701std::time_get< _CharT, _InIter > Class Template Reference	3290
5.701.1 Detailed Description	3292
5.701.2 Member Typedef Documentation	3292
5.701.2.1 char_type	3292
5.701.2.2 iter_type	3292
5.701.3 Constructor & Destructor Documentation	3293
5.701.3.1 time_get	3293
5.701.3.2 ~time_get	3293
5.701.4 Member Function Documentation	3293
5.701.4.1 date_order	3293
5.701.4.2 do_date_order	3293
5.701.4.3 do_get_date	3294
5.701.4.4 do_get_monthname	3294
5.701.4.5 do_get_time	3295
5.701.4.6 do_get_weekday	3296
5.701.4.7 do_get_year	3296

5.701.4.8	get_date	3297
5.701.4.9	get_monthname	3298
5.701.4.10	get_time	3298
5.701.4.11	lget_weekday	3299
5.701.4.12	get_year	3300
5.701.5	Member Data Documentation	3300
5.701.5.1	id	3300
5.702	std::time_get_byname<_CharT, _InIter > Class Template Reference	3301
5.702.1	Detailed Description	3303
5.702.2	Member Typedef Documentation	3303
5.702.2.1	char_type	3303
5.702.2.2	iter_type	3303
5.702.3	Member Function Documentation	3303
5.702.3.1	date_order	3303
5.702.3.2	do_date_order	3304
5.702.3.3	do_get_date	3304
5.702.3.4	do_get_monthname	3305
5.702.3.5	do_get_time	3305
5.702.3.6	do_get_weekday	3306
5.702.3.7	do_get_year	3307
5.702.3.8	get_date	3307
5.702.3.9	get_monthname	3308
5.702.3.10	get_time	3309
5.702.3.11	lget_weekday	3309
5.702.3.12	get_year	3310
5.702.4	Member Data Documentation	3310
5.702.4.1	id	3310
5.703	std::time_put<_CharT, _OutIter > Class Template Reference	3312
5.703.1	Detailed Description	3313
5.703.2	Member Typedef Documentation	3313
5.703.2.1	char_type	3313

5.703.2.2 iter_type	3314
5.703.3 Constructor & Destructor Documentation	3314
5.703.3.1 time_put	3314
5.703.3.2 ~time_put	3314
5.703.4 Member Function Documentation	3314
5.703.4.1 do_put	3314
5.703.4.2 put	3315
5.703.4.3 put	3316
5.703.5 Member Data Documentation	3316
5.703.5.1 id	3316
5.704std::time_put_byname< _CharT, _OutIter > Class Template Reference	3317
5.704.1 Detailed Description	3318
5.704.2 Member Typedef Documentation	3318
5.704.2.1 char_type	3318
5.704.2.2 iter_type	3318
5.704.3 Member Function Documentation	3319
5.704.3.1 do_put	3319
5.704.3.2 put	3319
5.704.3.3 put	3320
5.704.4 Member Data Documentation	3321
5.704.4.1 id	3321
5.705std::timed_mutex Class Reference	3322
5.705.1 Detailed Description	3322
5.706std::try_to_lock_t Struct Reference	3323
5.706.1 Detailed Description	3323
5.707std::tuple< _Elements > Class Template Reference	3324
5.707.1 Detailed Description	3324
5.708std::tuple< _T1, _T2 > Class Template Reference	3326
5.708.1 Detailed Description	3326
5.709std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference	3328

5.709.1 Detailed Description	3328
5.710std::tuple_element< __i, tuple< _Head, _Tail...> > Struct Template Reference	3329
5.710.1 Detailed Description	3329
5.711std::tuple_size< tuple< _Elements...> > Struct Template Reference	3330
5.711.1 Detailed Description	3330
5.712std::type_info Class Reference	3331
5.712.1 Detailed Description	3331
5.712.2 Constructor & Destructor Documentation	3331
5.712.2.1 ~type_info	3331
5.712.3 Member Function Documentation	3332
5.712.3.1 name	3332
5.713std::unary_function< _Arg, _Result > Struct Template Reference	3334
5.713.1 Detailed Description	3335
5.713.2 Member Typedef Documentation	3335
5.713.2.1 argument_type	3335
5.713.2.2 result_type	3335
5.714std::unary_negate< _Predicate > Class Template Reference	3336
5.714.1 Detailed Description	3336
5.714.2 Member Typedef Documentation	3337
5.714.2.1 argument_type	3337
5.714.2.2 result_type	3337
5.715std::underflow_error Class Reference	3338
5.715.1 Detailed Description	3338
5.715.2 Member Function Documentation	3338
5.715.2.1 what	3338
5.716std::uniform_int_distribution< _IntType > Class Template Reference	3340
5.716.1 Detailed Description	3341
5.716.2 Member Typedef Documentation	3341
5.716.2.1 result_type	3341
5.716.3 Constructor & Destructor Documentation	3341

5.716.3.1 uniform_int_distribution	3341
5.716.4 Member Function Documentation	3341
5.716.4.1 max	3341
5.716.4.2 min	3342
5.716.4.3 operator()	3342
5.716.4.4 operator()	3342
5.716.4.5 param	3342
5.716.4.6 param	3343
5.716.4.7 reset	3343
5.717std::uniform_int_distribution< _IntType >::param_type Struct Reference	3344
5.717.1 Detailed Description	3344
5.718std::uniform_real_distribution< _RealType > Class Template Reference	3345
5.718.1 Detailed Description	3345
5.718.2 Member Typedef Documentation	3346
5.718.2.1 result_type	3346
5.718.3 Constructor & Destructor Documentation	3346
5.718.3.1 uniform_real_distribution	3346
5.718.4 Member Function Documentation	3346
5.718.4.1 max	3346
5.718.4.2 min	3346
5.718.4.3 param	3347
5.718.4.4 param	3347
5.718.4.5 reset	3347
5.719std::uniform_real_distribution< _RealType >::param_type Struct Reference	3348
5.719.1 Detailed Description	3348
5.720std::unique_lock< _Mutex > Class Template Reference	3349
5.720.1 Detailed Description	3350
5.721std::unique_ptr< _Tp, _Tp_Deleter > Class Template Reference	3351
5.721.1 Detailed Description	3352
5.722std::unique_ptr< _Tp[], _Tp_Deleter > Class Template Reference	3353

5.722.1 Detailed Description	3354
5.723std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3355
5.723.1 Detailed Description	3355
5.724std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3357
5.724.1 Detailed Description	3357
5.725std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3359
5.725.1 Detailed Description	3359
5.726std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3361
5.726.1 Detailed Description	3361
5.727std::valarray< _Tp > Class Template Reference	3363
5.727.1 Detailed Description	3365
5.727.2 Constructor & Destructor Documentation	3366
5.727.2.1 valarray	3366
5.728std::vector< _Tp, _Alloc > Class Template Reference	3367
5.728.1 Detailed Description	3370
5.728.2 Constructor & Destructor Documentation	3371
5.728.2.1 vector	3371
5.728.2.2 vector	3371
5.728.2.3 vector	3371
5.728.2.4 vector	3371
5.728.2.5 vector	3372
5.728.2.6 vector	3372
5.728.2.7 vector	3372
5.728.2.8 ~vector	3373
5.728.3 Member Function Documentation	3373
5.728.3.1 _M_allocate_and_copy	3373
5.728.3.2 _M_range_check	3373
5.728.3.3 assign	3374

5.728.3.4 assign	3374
5.728.3.5 assign	3374
5.728.3.6 at	3375
5.728.3.7 at	3375
5.728.3.8 back	3376
5.728.3.9 back	3376
5.728.3.10begin	3376
5.728.3.11begin	3376
5.728.3.12capacity	3377
5.728.3.13begin	3377
5.728.3.14end	3377
5.728.3.15clear	3377
5.728.3.16crbegin	3378
5.728.3.17crend	3378
5.728.3.18data	3378
5.728.3.19emplace	3378
5.728.3.20empty	3379
5.728.3.21end	3379
5.728.3.22end	3379
5.728.3.23erase	3379
5.728.3.24erase	3380
5.728.3.25front	3380
5.728.3.26front	3381
5.728.3.27insert	3381
5.728.3.28insert	3381
5.728.3.29insert	3382
5.728.3.30insert	3382
5.728.3.31insert	3383
5.728.3.32max_size	3383
5.728.3.33operator=	3383
5.728.3.34operator=	3384

5.728.3.35	operator=	3384
5.728.3.36	operator[]	3384
5.728.3.37	operator[]	3385
5.728.3.38	pop_back	3385
5.728.3.39	push_back	3385
5.728.3.40	begin	3386
5.728.3.41	rbegin	3386
5.728.3.42	end	3386
5.728.3.43	rend	3386
5.728.3.44	reserve	3387
5.728.3.45	resize	3387
5.728.3.46	shrink_to_fit	3388
5.728.3.47	size	3388
5.728.3.48	swap	3388
5.729	std::vector< bool, _Alloc > Class Template Reference	3389
5.729.1	Detailed Description	3392
5.730	std::weak_ptr< _Tp > Class Template Reference	3394
5.730.1	Detailed Description	3394
5.731	std::weibull_distribution< _RealType > Class Template Reference	3395
5.731.1	Detailed Description	3395
5.731.2	Member Typedef Documentation	3396
5.731.2.1	result_type	3396
5.731.3	Member Function Documentation	3396
5.731.3.1	a	3396
5.731.3.2	b	3396
5.731.3.3	max	3396
5.731.3.4	min	3396
5.731.3.5	param	3396
5.731.3.6	param	3397
5.731.3.7	reset	3397
5.732	std::weibull_distribution< _RealType >::param_type Struct Reference	3398

5.732.1 Detailed Description	3398
6 File Documentation	3399
6.1 algo.h File Reference	3399
6.1.1 Detailed Description	3413
6.2 algbase.h File Reference	3414
6.2.1 Detailed Description	3415
6.3 algorithm File Reference	3416
6.3.1 Detailed Description	3416
6.4 algorithm File Reference	3417
6.4.1 Detailed Description	3418
6.5 algorithm File Reference	3419
6.5.1 Detailed Description	3419
6.6 algorithmfwd.h File Reference	3420
6.6.1 Detailed Description	3426
6.7 algorithmfwd.h File Reference	3427
6.7.1 Detailed Description	3439
6.8 allocator.h File Reference	3440
6.8.1 Detailed Description	3440
6.9 array File Reference	3441
6.9.1 Detailed Description	3441
6.10 array File Reference	3442
6.10.1 Detailed Description	3442
6.11 array_allocator.h File Reference	3444
6.11.1 Detailed Description	3444
6.12 assoc_container.hpp File Reference	3445
6.12.1 Detailed Description	3446
6.13 atomic File Reference	3447
6.13.1 Detailed Description	3451
6.14 atomic_0.h File Reference	3452
6.14.1 Detailed Description	3452

6.15	atomic_2.h File Reference	3453
6.15.1	Detailed Description	3453
6.16	atomic_base.h File Reference	3454
6.16.1	Detailed Description	3455
6.17	atomic_word.h File Reference	3456
6.17.1	Detailed Description	3456
6.18	atomicfwd_c.h File Reference	3457
6.18.1	Detailed Description	3458
6.19	atomicfwd_cxx.h File Reference	3459
6.19.1	Detailed Description	3459
6.20	atomicity.h File Reference	3460
6.20.1	Detailed Description	3460
6.21	auto_ptr.h File Reference	3461
6.21.1	Detailed Description	3461
6.22	balanced_quicksort.h File Reference	3462
6.22.1	Detailed Description	3462
6.23	base.h File Reference	3464
6.23.1	Detailed Description	3465
6.24	base.h File Reference	3466
6.24.1	Detailed Description	3466
6.25	basic_file.h File Reference	3467
6.25.1	Detailed Description	3467
6.26	basic_ios.h File Reference	3468
6.26.1	Detailed Description	3468
6.27	basic_ios.tcc File Reference	3469
6.27.1	Detailed Description	3469
6.28	basic_iterator.h File Reference	3470
6.28.1	Detailed Description	3470
6.29	basic_string.h File Reference	3471
6.29.1	Detailed Description	3474
6.30	basic_string.tcc File Reference	3475

6.30.1 Detailed Description	3476
6.31 basic_types.hpp File Reference	3477
6.31.1 Detailed Description	3477
6.32 binders.h File Reference	3478
6.32.1 Detailed Description	3478
6.33 bitmap_allocator.h File Reference	3479
6.33.1 Detailed Description	3480
6.33.2 Define Documentation	3480
6.33.2.1 _BALLOC_ALIGN_BYTES	3480
6.34 bitset File Reference	3481
6.34.1 Detailed Description	3482
6.35 bitset File Reference	3483
6.35.1 Detailed Description	3483
6.36 bitset File Reference	3484
6.36.1 Detailed Description	3484
6.37 boost_concept_check.h File Reference	3485
6.37.1 Detailed Description	3485
6.38 boost_sp_counted_base.h File Reference	3486
6.38.1 Detailed Description	3486
6.39 c++0x_warning.h File Reference	3487
6.39.1 Detailed Description	3487
6.40 c++allocator.h File Reference	3488
6.40.1 Detailed Description	3488
6.41 c++config.h File Reference	3489
6.41.1 Detailed Description	3493
6.42 c++io.h File Reference	3494
6.42.1 Detailed Description	3494
6.43 c++locale.h File Reference	3495
6.43.1 Detailed Description	3495
6.44 c++locale_internal.h File Reference	3496
6.44.1 Detailed Description	3496

6.45	cassert File Reference	3497
6.45.1	Detailed Description	3497
6.46	ccomplex File Reference	3498
6.46.1	Detailed Description	3498
6.47	ccomplex File Reference	3499
6.47.1	Detailed Description	3499
6.48	cctype File Reference	3500
6.48.1	Detailed Description	3500
6.49	cctype File Reference	3501
6.49.1	Detailed Description	3501
6.50	cctype File Reference	3502
6.50.1	Detailed Description	3502
6.51	cerrno File Reference	3503
6.51.1	Detailed Description	3503
6.52	cfenv File Reference	3504
6.52.1	Detailed Description	3504
6.53	cfenv File Reference	3505
6.53.1	Detailed Description	3505
6.54	cfenv File Reference	3506
6.54.1	Detailed Description	3506
6.55	cfloat File Reference	3507
6.55.1	Detailed Description	3507
6.56	cfloat File Reference	3508
6.56.1	Detailed Description	3508
6.57	char_traits.h File Reference	3509
6.57.1	Detailed Description	3509
6.58	checkers.h File Reference	3510
6.58.1	Detailed Description	3510
6.59	chrono File Reference	3511
6.59.1	Detailed Description	3514
6.60	cinttypes File Reference	3515

6.60.1 Detailed Description	3515
6.61 cinttypes File Reference	3516
6.61.1 Detailed Description	3516
6.62 cinttypes File Reference	3517
6.62.1 Detailed Description	3517
6.63 ciso646 File Reference	3518
6.63.1 Detailed Description	3518
6.64 climits File Reference	3519
6.64.1 Detailed Description	3519
6.65 climits File Reference	3520
6.65.1 Detailed Description	3520
6.66 clocale File Reference	3521
6.66.1 Detailed Description	3521
6.67 cmath File Reference	3522
6.67.1 Detailed Description	3525
6.68 cmath File Reference	3526
6.68.1 Detailed Description	3529
6.69 cmath File Reference	3530
6.69.1 Detailed Description	3530
6.70 cmath.tcc File Reference	3531
6.70.1 Detailed Description	3531
6.71 codecvt.h File Reference	3532
6.71.1 Detailed Description	3533
6.72 codecvt_specializations.h File Reference	3534
6.72.1 Detailed Description	3534
6.73 compatibility.h File Reference	3535
6.73.1 Detailed Description	3535
6.74 compatibility.h File Reference	3536
6.74.1 Detailed Description	3536
6.75 compiletime_settings.h File Reference	3537
6.75.1 Detailed Description	3537

6.75.2	Define Documentation	3537
6.75.2.1	_GLIBCXX_ASSERTIONS	3537
6.75.2.2	_GLIBCXX_CALL	3537
6.75.2.3	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_L1	3538
6.75.2.4	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_TLB	3538
6.75.2.5	_GLIBCXX_SCALE_DOWN_FPU	3538
6.75.2.6	_GLIBCXX_VERBOSE_LEVEL	3538
6.76	complex File Reference	3539
6.76.1	Detailed Description	3542
6.77	complex File Reference	3543
6.77.1	Detailed Description	3543
6.78	complex File Reference	3544
6.78.1	Detailed Description	3545
6.79	complex.h File Reference	3546
6.79.1	Detailed Description	3546
6.80	concept_check.h File Reference	3547
6.80.1	Detailed Description	3547
6.81	concurrency.h File Reference	3548
6.81.1	Detailed Description	3548
6.82	cond_dealtor.hpp File Reference	3549
6.82.1	Detailed Description	3549
6.83	condition_variable File Reference	3550
6.83.1	Detailed Description	3550
6.84	constructors_destructor_fn_imps.hpp File Reference	3551
6.84.1	Detailed Description	3551
6.85	container_base_dispatch.hpp File Reference	3552
6.85.1	Detailed Description	3552
6.86	cpp_type_traits.h File Reference	3553
6.86.1	Detailed Description	3553
6.87	cpu_defines.h File Reference	3554

6.87.1 Detailed Description	3554
6.88 csetjmp File Reference	3555
6.88.1 Detailed Description	3555
6.89 csignal File Reference	3556
6.89.1 Detailed Description	3556
6.90 cstdarg File Reference	3557
6.90.1 Detailed Description	3557
6.91 cstdarg File Reference	3558
6.91.1 Detailed Description	3558
6.92 cstdbool File Reference	3559
6.92.1 Detailed Description	3559
6.93 cstdbool File Reference	3560
6.93.1 Detailed Description	3560
6.94 cstddef File Reference	3561
6.94.1 Detailed Description	3561
6.95 cstdint File Reference	3562
6.95.1 Detailed Description	3562
6.96 cstdint File Reference	3563
6.96.1 Detailed Description	3563
6.97 cstdint File Reference	3564
6.97.1 Detailed Description	3564
6.98 cstdio File Reference	3565
6.98.1 Detailed Description	3565
6.99 cstdio File Reference	3566
6.99.1 Detailed Description	3566
6.100 cstdio File Reference	3567
6.100.1 Detailed Description	3567
6.101 cstdlib File Reference	3568
6.101.1 Detailed Description	3568
6.102 cstdlib File Reference	3569
6.102.1 Detailed Description	3569

6.103cstdlib File Reference	3570
6.103.1 Detailed Description	3570
6.104cstring File Reference	3571
6.104.1 Detailed Description	3571
6.105ctgmth File Reference	3572
6.105.1 Detailed Description	3572
6.106ctgmth File Reference	3573
6.106.1 Detailed Description	3573
6.107ctime File Reference	3574
6.107.1 Detailed Description	3574
6.108ctime File Reference	3575
6.108.1 Detailed Description	3575
6.109ctype_base.h File Reference	3576
6.109.1 Detailed Description	3576
6.110ctype_inline.h File Reference	3577
6.110.1 Detailed Description	3577
6.111ctype_noninline.h File Reference	3578
6.111.1 Detailed Description	3578
6.112cwchar File Reference	3579
6.112.1 Detailed Description	3579
6.113cwchar File Reference	3580
6.113.1 Detailed Description	3580
6.114cwchar File Reference	3581
6.114.1 Detailed Description	3581
6.115cwctype File Reference	3582
6.115.1 Detailed Description	3582
6.116cwctype File Reference	3583
6.116.1 Detailed Description	3583
6.117cwctype File Reference	3584
6.117.1 Detailed Description	3584
6.118cxxabi-forced.h File Reference	3585

6.118.1 Detailed Description	3585
6.119cxxabi.h File Reference	3586
6.119.1 Detailed Description	3587
6.120cxxabi_tweaks.h File Reference	3588
6.120.1 Detailed Description	3588
6.121debug.h File Reference	3589
6.121.1 Detailed Description	3589
6.122debug_allocator.h File Reference	3590
6.122.1 Detailed Description	3590
6.123debug_map_base.hpp File Reference	3591
6.123.1 Detailed Description	3591
6.124decimal File Reference	3592
6.124.1 Detailed Description	3603
6.125deque File Reference	3604
6.125.1 Detailed Description	3604
6.126deque File Reference	3605
6.126.1 Detailed Description	3606
6.127deque File Reference	3607
6.127.1 Detailed Description	3608
6.128deque.tcc File Reference	3609
6.128.1 Detailed Description	3609
6.129enc_filebuf.h File Reference	3610
6.129.1 Detailed Description	3610
6.130equally_split.h File Reference	3611
6.130.1 Detailed Description	3611
6.131error_constants.h File Reference	3612
6.131.1 Detailed Description	3613
6.132exception File Reference	3614
6.132.1 Detailed Description	3614
6.133exception.hpp File Reference	3615
6.133.1 Detailed Description	3615

6.134	exception_ptr.h File Reference	3616
6.134.1	Detailed Description	3616
6.135	extptr_allocator.h File Reference	3617
6.135.1	Detailed Description	3617
6.136	features.h File Reference	3618
6.136.1	Detailed Description	3618
6.136.2	Define Documentation	3618
6.136.2.1	_GLIBCXX_BAL_QUICKSORT	3618
6.136.2.2	_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS	3618
6.136.2.3	_GLIBCXX_FIND_EQUAL_SPLIT	3619
6.136.2.4	_GLIBCXX_FIND_GROWING_BLOCKS	3619
6.136.2.5	_GLIBCXX_MERGESORT	3619
6.136.2.6	_GLIBCXX_QUICKSORT	3619
6.136.2.7	_GLIBCXX_TREE_DYNAMIC_BALANCING	3619
6.136.2.8	_GLIBCXX_TREE_FULL_COPY	3620
6.136.2.9	_GLIBCXX_TREE_INITIAL_SPLITTING	3620
6.137	fenv.h File Reference	3621
6.137.1	Detailed Description	3621
6.138	find.h File Reference	3622
6.138.1	Detailed Description	3622
6.139	find_selectors.h File Reference	3623
6.139.1	Detailed Description	3623
6.140	for_each.h File Reference	3624
6.140.1	Detailed Description	3624
6.141	for_each_selectors.h File Reference	3625
6.141.1	Detailed Description	3626
6.142	formatter.h File Reference	3627
6.142.1	Detailed Description	3628
6.143	forward_list.h File Reference	3629
6.143.1	Detailed Description	3630
6.144	forward_list.tcc File Reference	3631

6.144.1 Detailed Description	3631
6.145fstream File Reference	3632
6.145.1 Detailed Description	3632
6.146fstream.tcc File Reference	3633
6.146.1 Detailed Description	3633
6.147functexcept.h File Reference	3634
6.147.1 Detailed Description	3634
6.148functional File Reference	3635
6.148.1 Detailed Description	3639
6.149functional File Reference	3640
6.149.1 Detailed Description	3641
6.150functional_hash.h File Reference	3642
6.150.1 Detailed Description	3642
6.151functions.h File Reference	3643
6.151.1 Detailed Description	3645
6.152future File Reference	3646
6.152.1 Detailed Description	3648
6.153gslice.h File Reference	3649
6.153.1 Detailed Description	3649
6.154gslice_array.h File Reference	3650
6.154.1 Detailed Description	3650
6.155hash_fun.h File Reference	3651
6.155.1 Detailed Description	3651
6.156hash_map File Reference	3652
6.156.1 Detailed Description	3653
6.157hash_policy.hpp File Reference	3654
6.157.1 Detailed Description	3654
6.158hash_set File Reference	3655
6.158.1 Detailed Description	3656
6.159hashtable.h File Reference	3657
6.159.1 Detailed Description	3657

6.160	hashtable.h File Reference	3658
6.160.1	Detailed Description	3658
6.161	hashtable_policy.h File Reference	3659
6.161.1	Detailed Description	3660
6.162	indirect_array.h File Reference	3661
6.162.1	Detailed Description	3661
6.163	initializer_list File Reference	3662
6.163.1	Detailed Description	3662
6.164	iomanip File Reference	3663
6.164.1	Detailed Description	3664
6.165	ios File Reference	3665
6.165.1	Detailed Description	3665
6.166	ios_base.h File Reference	3666
6.166.1	Detailed Description	3668
6.167	iosfwd File Reference	3669
6.167.1	Detailed Description	3669
6.168	iostream File Reference	3670
6.168.1	Detailed Description	3670
6.169	istream File Reference	3671
6.169.1	Detailed Description	3672
6.170	istream.tcc File Reference	3673
6.170.1	Detailed Description	3673
6.171	iterator File Reference	3674
6.171.1	Detailed Description	3674
6.172	iterator File Reference	3675
6.172.1	Detailed Description	3675
6.173	iterator.h File Reference	3676
6.173.1	Detailed Description	3676
6.174	iterator_tracker.h File Reference	3677
6.174.1	Detailed Description	3678
6.175	limits File Reference	3679

6.175.1 Detailed Description	3681
6.176list File Reference	3682
6.176.1 Detailed Description	3682
6.177list File Reference	3683
6.177.1 Detailed Description	3684
6.178list File Reference	3685
6.178.1 Detailed Description	3686
6.179list.tcc File Reference	3687
6.179.1 Detailed Description	3687
6.180list_partition.h File Reference	3688
6.180.1 Detailed Description	3688
6.181list_update_policy.hpp File Reference	3689
6.181.1 Detailed Description	3689
6.182locale File Reference	3690
6.182.1 Detailed Description	3690
6.183locale_classes.h File Reference	3691
6.183.1 Detailed Description	3692
6.184locale_classes.tcc File Reference	3693
6.184.1 Detailed Description	3693
6.185locale_facets.h File Reference	3694
6.185.1 Detailed Description	3696
6.186locale_facets.tcc File Reference	3697
6.186.1 Detailed Description	3698
6.187locale_facets_nonio.h File Reference	3699
6.187.1 Detailed Description	3700
6.188locale_facets_nonio.tcc File Reference	3701
6.188.1 Detailed Description	3702
6.189localefwd.h File Reference	3703
6.189.1 Detailed Description	3704
6.190losertree.h File Reference	3705
6.190.1 Detailed Description	3706

6.191 macros.h File Reference	3707
6.191.1 Detailed Description	3707
6.191.2 Define Documentation	3707
6.191.2.1 __glibcxx_check_erase	3707
6.191.2.2 __glibcxx_check_erase_range	3708
6.191.2.3 __glibcxx_check_heap_pred	3708
6.191.2.4 __glibcxx_check_insert	3708
6.191.2.5 __glibcxx_check_insert_range	3708
6.191.2.6 __glibcxx_check_partitioned_lower	3708
6.191.2.7 __glibcxx_check_partitioned_lower_pred	3709
6.191.2.8 __glibcxx_check_partitioned_upper_pred	3709
6.191.2.9 __glibcxx_check_sorted_pred	3709
6.191.2.10 GLIBCXX_DEBUG_VERIFY	3709
6.192 malloc_allocator.h File Reference	3710
6.192.1 Detailed Description	3710
6.193 map File Reference	3711
6.193.1 Detailed Description	3711
6.194 map File Reference	3712
6.194.1 Detailed Description	3712
6.195 map File Reference	3713
6.195.1 Detailed Description	3713
6.196 map.h File Reference	3714
6.196.1 Detailed Description	3715
6.197 map.h File Reference	3716
6.197.1 Detailed Description	3717
6.198 mask_array.h File Reference	3718
6.198.1 Detailed Description	3718
6.199 memory File Reference	3719
6.199.1 Detailed Description	3719
6.200 memory File Reference	3720
6.200.1 Detailed Description	3720

6.201	merge.h File Reference	3722
6.201.1	Detailed Description	3723
6.202	messages_members.h File Reference	3724
6.202.1	Detailed Description	3724
6.203	move.h File Reference	3725
6.203.1	Detailed Description	3726
6.204	mt_allocator.h File Reference	3727
6.204.1	Detailed Description	3728
6.205	multimap.h File Reference	3729
6.205.1	Detailed Description	3730
6.206	multimap.h File Reference	3731
6.206.1	Detailed Description	3732
6.207	multiseq_selection.h File Reference	3733
6.207.1	Detailed Description	3733
6.208	multiset.h File Reference	3735
6.208.1	Detailed Description	3736
6.209	multiset.h File Reference	3737
6.209.1	Detailed Description	3738
6.210	multiway_merge.h File Reference	3739
6.210.1	Detailed Description	3744
6.210.2	Define Documentation	3744
6.210.2.1	_GLIBCXX_PARALLEL_LENGTH	3744
6.211	multiway_mergesort.h File Reference	3745
6.211.1	Detailed Description	3746
6.212	mutex File Reference	3747
6.212.1	Detailed Description	3748
6.213	nested_exception.h File Reference	3749
6.213.1	Detailed Description	3749
6.214	new File Reference	3750
6.214.1	Detailed Description	3751
6.214.2	Function Documentation	3751

6.214.2.1 operator delete	3751
6.214.2.2 operator delete	3751
6.214.2.3 operator delete	3752
6.214.2.4 operator delete[]	3752
6.214.2.5 operator delete[]	3752
6.214.2.6 operator delete[]	3753
6.214.2.7 operator new	3753
6.214.2.8 operator new	3753
6.214.2.9 operator new	3754
6.214.2.10operator new[]	3754
6.214.2.11operator new[]	3754
6.214.2.12operator new[]	3755
6.215new_allocator.h File Reference	3756
6.215.1 Detailed Description	3756
6.216numeric File Reference	3757
6.216.1 Detailed Description	3757
6.217numeric File Reference	3758
6.217.1 Detailed Description	3758
6.218numeric File Reference	3759
6.218.1 Detailed Description	3762
6.219numeric_traits.h File Reference	3763
6.219.1 Detailed Description	3763
6.220numeric_fwd.h File Reference	3764
6.220.1 Detailed Description	3766
6.221omp_loop.h File Reference	3767
6.221.1 Detailed Description	3767
6.222omp_loop_static.h File Reference	3768
6.222.1 Detailed Description	3768
6.223os_defines.h File Reference	3769
6.223.1 Detailed Description	3769
6.224ostream File Reference	3770

6.224.1 Detailed Description	3771
6.225 ostream.tcc File Reference	3772
6.225.1 Detailed Description	3772
6.226 ostream_insert.h File Reference	3773
6.226.1 Detailed Description	3773
6.227 par_loop.h File Reference	3774
6.227.1 Detailed Description	3774
6.228 parallel.h File Reference	3775
6.228.1 Detailed Description	3775
6.229 partial_sum.h File Reference	3776
6.229.1 Detailed Description	3776
6.230 partition.h File Reference	3777
6.230.1 Detailed Description	3777
6.230.2 Define Documentation	3777
6.230.2.1 _GLIBCXX_VOLATILE	3777
6.231 pod_char_traits.h File Reference	3778
6.231.1 Detailed Description	3778
6.232 pointer.h File Reference	3779
6.232.1 Detailed Description	3781
6.233 pool_allocator.h File Reference	3782
6.233.1 Detailed Description	3782
6.234 postypes.h File Reference	3783
6.234.1 Detailed Description	3783
6.235 priority_queue.hpp File Reference	3784
6.235.1 Detailed Description	3784
6.236 priority_queue_base_dispatch.hpp File Reference	3785
6.236.1 Detailed Description	3785
6.237 profiler.h File Reference	3786
6.237.1 Detailed Description	3789
6.238 profiler_hashtable_size.h File Reference	3790
6.238.1 Detailed Description	3790

6.239	profiler_list_to_slist.h File Reference	3791
6.239.1	Detailed Description	3791
6.240	profiler_list_to_vector.h File Reference	3792
6.240.1	Detailed Description	3792
6.241	profiler_map_to_unordered_map.h File Reference	3793
6.241.1	Detailed Description	3794
6.242	profiler_node.h File Reference	3795
6.242.1	Detailed Description	3796
6.243	profiler_state.h File Reference	3797
6.243.1	Detailed Description	3797
6.244	profiler_trace.h File Reference	3798
6.244.1	Detailed Description	3800
6.245	profiler_vector_size.h File Reference	3801
6.245.1	Detailed Description	3801
6.246	profiler_vector_to_list.h File Reference	3802
6.246.1	Detailed Description	3803
6.247	queue File Reference	3804
6.247.1	Detailed Description	3804
6.248	queue.h File Reference	3805
6.248.1	Detailed Description	3805
6.248.2	Define Documentation	3805
6.248.2.1	_GLIBCXX_VOLATILE	3805
6.249	quicksort.h File Reference	3806
6.249.1	Detailed Description	3806
6.250	random File Reference	3807
6.250.1	Detailed Description	3807
6.251	random.h File Reference	3808
6.251.1	Detailed Description	3812
6.252	random.tcc File Reference	3813
6.252.1	Detailed Description	3817
6.253	random_number.h File Reference	3818

6.253.1 Detailed Description	3818
6.254random_shuffle.h File Reference	3819
6.254.1 Detailed Description	3820
6.255ratio File Reference	3821
6.255.1 Detailed Description	3822
6.256rb_tree File Reference	3823
6.256.1 Detailed Description	3823
6.257rc_string_base.h File Reference	3824
6.257.1 Detailed Description	3824
6.258regex File Reference	3825
6.258.1 Detailed Description	3825
6.259regex File Reference	3826
6.259.1 Detailed Description	3834
6.260rope File Reference	3835
6.260.1 Detailed Description	3839
6.261ropeimpl.h File Reference	3840
6.261.1 Detailed Description	3840
6.262safe_base.h File Reference	3841
6.262.1 Detailed Description	3841
6.263safe_iterator.h File Reference	3842
6.263.1 Detailed Description	3843
6.264safe_iterator.tcc File Reference	3844
6.264.1 Detailed Description	3844
6.265safe_sequence.h File Reference	3845
6.265.1 Detailed Description	3845
6.266search.h File Reference	3846
6.266.1 Detailed Description	3846
6.267set File Reference	3847
6.267.1 Detailed Description	3847
6.268set File Reference	3848
6.268.1 Detailed Description	3848

6.269set File Reference	3849
6.269.1 Detailed Description	3849
6.270set.h File Reference	3850
6.270.1 Detailed Description	3851
6.271set.h File Reference	3852
6.271.1 Detailed Description	3853
6.272set_operations.h File Reference	3854
6.272.1 Detailed Description	3854
6.273settings.h File Reference	3856
6.273.1 Detailed Description	3856
6.273.2 parallelization_decision	3856
6.273.3 Define Documentation	3857
6.273.3.1 _GLIBCXX_PARALLEL_CONDITION	3857
6.274shared_ptr.h File Reference	3858
6.274.1 Detailed Description	3859
6.275shared_ptr_base.h File Reference	3860
6.275.1 Detailed Description	3860
6.276slice_array.h File Reference	3861
6.276.1 Detailed Description	3861
6.277slist File Reference	3862
6.277.1 Detailed Description	3863
6.278sort.h File Reference	3864
6.278.1 Detailed Description	3865
6.279sso_string_base.h File Reference	3866
6.279.1 Detailed Description	3866
6.280sstream File Reference	3867
6.280.1 Detailed Description	3867
6.281sstream.tcc File Reference	3868
6.281.1 Detailed Description	3868
6.282stack File Reference	3869
6.282.1 Detailed Description	3869

6.283	<code>standard_policies.hpp</code> File Reference	3870
6.283.1	Detailed Description	3870
6.284	<code>stdatomic.h</code> File Reference	3871
6.284.1	Detailed Description	3871
6.285	<code>stdexcept</code> File Reference	3872
6.285.1	Detailed Description	3872
6.286	<code>stdio_filebuf.h</code> File Reference	3873
6.286.1	Detailed Description	3873
6.287	<code>stdio_sync_filebuf.h</code> File Reference	3874
6.287.1	Detailed Description	3874
6.288	<code>stl_algo.h</code> File Reference	3875
6.288.1	Detailed Description	3888
6.289	<code>stl_algobase.h</code> File Reference	3889
6.289.1	Detailed Description	3891
6.290	<code>stl_bvector.h</code> File Reference	3892
6.290.1	Detailed Description	3893
6.291	<code>stl_construct.h</code> File Reference	3894
6.291.1	Detailed Description	3894
6.292	<code>stl_deque.h</code> File Reference	3895
6.292.1	Detailed Description	3898
6.292.2	Define Documentation	3898
6.292.2.1	<code>_GLIBCXX_DEQUE_BUF_SIZE</code>	3898
6.293	<code>stl_function.h</code> File Reference	3899
6.293.1	Detailed Description	3901
6.294	<code>stl_heap.h</code> File Reference	3902
6.294.1	Detailed Description	3904
6.295	<code>stl_iterator.h</code> File Reference	3905
6.295.1	Detailed Description	3908
6.296	<code>stl_iterator_base_funcs.h</code> File Reference	3909
6.296.1	Detailed Description	3909
6.297	<code>stl_iterator_base_types.h</code> File Reference	3911

6.297.1 Detailed Description	3912
6.298stl_list.h File Reference	3913
6.298.1 Detailed Description	3914
6.299stl_map.h File Reference	3915
6.299.1 Detailed Description	3916
6.300stl_multimap.h File Reference	3917
6.300.1 Detailed Description	3918
6.301stl_multiset.h File Reference	3919
6.301.1 Detailed Description	3920
6.302stl_numeric.h File Reference	3921
6.302.1 Detailed Description	3922
6.303stl_pair.h File Reference	3923
6.303.1 Detailed Description	3924
6.304stl_queue.h File Reference	3925
6.304.1 Detailed Description	3926
6.305stl_raw_storage_iter.h File Reference	3927
6.305.1 Detailed Description	3927
6.306stl_relops.h File Reference	3928
6.306.1 Detailed Description	3928
6.307stl_set.h File Reference	3929
6.307.1 Detailed Description	3930
6.308stl_stack.h File Reference	3931
6.308.1 Detailed Description	3932
6.309stl_tempbuf.h File Reference	3933
6.309.1 Detailed Description	3933
6.310stl_tree.h File Reference	3934
6.310.1 Detailed Description	3935
6.311stl_uninitialized.h File Reference	3936
6.311.1 Detailed Description	3937
6.312stl_vector.h File Reference	3938
6.312.1 Detailed Description	3939

6.313	stream_iterator.h File Reference	3940
6.313.1	Detailed Description	3940
6.314	streambuf File Reference	3941
6.314.1	Detailed Description	3941
6.315	streambuf.tcc File Reference	3942
6.315.1	Detailed Description	3942
6.316	streambuf_iterator.h File Reference	3943
6.316.1	Detailed Description	3944
6.317	string File Reference	3945
6.317.1	Detailed Description	3945
6.318	string File Reference	3946
6.318.1	Detailed Description	3948
6.319	stringfwd.h File Reference	3949
6.319.1	Detailed Description	3949
6.320	system_error File Reference	3950
6.320.1	Detailed Description	3951
6.321	tag_and_trait.hpp File Reference	3952
6.321.1	Detailed Description	3954
6.322	tags.h File Reference	3955
6.322.1	Detailed Description	3956
6.323	tgmath.h File Reference	3957
6.323.1	Detailed Description	3957
6.324	thread File Reference	3958
6.324.1	Detailed Description	3959
6.325	throw_allocator.h File Reference	3960
6.325.1	Detailed Description	3962
6.326	time_members.h File Reference	3963
6.326.1	Detailed Description	3963
6.327	tree_policy.hpp File Reference	3964
6.327.1	Detailed Description	3964
6.328	tree_trace_base.hpp File Reference	3965

6.328.1 Detailed Description	3965
6.329trie_policy.hpp File Reference	3966
6.329.1 Detailed Description	3966
6.330tuple File Reference	3967
6.330.1 Detailed Description	3969
6.331type_traits File Reference	3970
6.331.1 Detailed Description	3972
6.332type_traits File Reference	3973
6.332.1 Detailed Description	3976
6.333type_traits.h File Reference	3977
6.333.1 Detailed Description	3977
6.334type_utils.hpp File Reference	3978
6.334.1 Detailed Description	3978
6.335typeinfo File Reference	3979
6.335.1 Detailed Description	3979
6.336typelist.h File Reference	3980
6.336.1 Detailed Description	3981
6.337types.h File Reference	3982
6.337.1 Detailed Description	3983
6.338types_traits.hpp File Reference	3984
6.338.1 Detailed Description	3984
6.339unique_copy.h File Reference	3985
6.339.1 Detailed Description	3985
6.340unique_ptr.h File Reference	3986
6.340.1 Detailed Description	3987
6.341unordered_map File Reference	3988
6.341.1 Detailed Description	3988
6.342unordered_map File Reference	3989
6.342.1 Detailed Description	3989
6.343unordered_map File Reference	3990
6.343.1 Detailed Description	3990

6.344unordered_map.h File Reference	3991
6.344.1 Detailed Description	3991
6.345unordered_set File Reference	3992
6.345.1 Detailed Description	3992
6.346unordered_set File Reference	3993
6.346.1 Detailed Description	3993
6.347unordered_set File Reference	3994
6.347.1 Detailed Description	3994
6.348unordered_set.h File Reference	3995
6.348.1 Detailed Description	3995
6.349utility File Reference	3996
6.349.1 Detailed Description	3996
6.350utility File Reference	3997
6.350.1 Detailed Description	3997
6.351valarray File Reference	3998
6.351.1 Detailed Description	4003
6.352valarray_after.h File Reference	4004
6.352.1 Detailed Description	4018
6.353valarray_array.h File Reference	4019
6.353.1 Detailed Description	4029
6.354valarray_array.tcc File Reference	4030
6.354.1 Detailed Description	4031
6.355valarray_before.h File Reference	4032
6.355.1 Detailed Description	4032
6.356vector File Reference	4033
6.356.1 Detailed Description	4033
6.357vector File Reference	4034
6.357.1 Detailed Description	4035
6.358vector File Reference	4036
6.358.1 Detailed Description	4037
6.359vector.tcc File Reference	4038

6.359.1 Detailed Description	4038
6.360vstring.h File Reference	4039
6.360.1 Detailed Description	4043
6.361vstring.tcc File Reference	4044
6.361.1 Detailed Description	4045
6.362vstring_fwd.h File Reference	4046
6.362.1 Detailed Description	4046
6.363vstring_util.h File Reference	4047
6.363.1 Detailed Description	4047
6.364workstealing.h File Reference	4048
6.364.1 Detailed Description	4048

Chapter 1

Todo List

Member [__glibcxx_check_insert_range](#) We would like to be able to check for non-interference 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)`
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::hash_map](#)`<_Key, _Tp, _HashFn, _EqualKey, _Alloc >`
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::hash_multimap](#)`<_Key, _Tp, _HashFn, _EqualKey, _Alloc >`
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::hash_multiset](#)`<_Value, _HashFn, _EqualKey, _Alloc >`
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::iota\(_ForwardIter __first, _ForwardIter __last, _Tp __value\)](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::power\(_Tp __x, _Integer __n\)](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample\(_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _Size __n\)](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::random_sample_n`(`ForwardIterator __first`, `ForwardIterator __last`, `OutputIterator __out`, `count`)

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class `__gnu_cxx::rope`< `_CharT`, `_Alloc` > Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class `__gnu_cxx::slist`< `_Tp`, `_Alloc` > Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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` >

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::discard_block_engine::discard`(`unsigned long long __z`) Look for a faster way to do discard.

Member `std::discard_block_engine::max`() `const` This should be `constexpr`.

Member `std::discard_block_engine::min`() `const` This should be `constexpr`.

Member `std::independent_bits_engine::discard(unsigned long long __z)` Look for a faster way to do discard.

Member `std::independent_bits_engine::max() const` This should be constexpr.

Member `std::independent_bits_engine::min() const` This should be constexpr.

Member `std::linear_congruential_engine::discard(unsigned long long __z)`
Look for a faster way to do discard.

Member `std::linear_congruential_engine::max() const` This should be constexpr.

Member `std::linear_congruential_engine::min() const` This should be constexpr.

Member `std::match_results::format(const string_type &__fmt, regex_constants::match_flag_type __flags=)`
Implement this function.
Implement this function.
Implement this function.

Member `std::operator==(const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_`
Implement this function.

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

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **std::regex_match**(`_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Allocator > &__m, const regex< _Ch_type, _Rx_traits, _String_allocator > &__r`)
 Implement this function.

Member **std::regex_replace**(`const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits, _String_allocator > &__r, const basic_string< _Ch_type > &__str`)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Member **std::regex_search**(`const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits, _String_allocator > &__r, const basic_string< _Ch_type > &__str`)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

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++(int)` Implement this function.
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()` Implement this function.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Implement this function.

Member `std::regex_traits::lookup_classname(_Fwd_iter __first, _Fwd_iter __last) 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`**
 Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator+(difference_type __n) const`**
 Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator++(int)`** Doc me! See `doc/doxygen/TODO`
 and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator+=(difference_type __n)`**
 Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator-(difference_type __n) const`**
 Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator--(int)`** Doc me! See `doc/doxygen/TODO`
 and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Doc me! See `doc/doxygen/TODO` and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::reverse_iterator::operator=(difference_type __n)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator::operator->() const`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator::operator[](difference_type __n) const`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::shuffle_order_engine::discard(unsigned long long __z)`** Look for a faster way to do discard.

Member **`std::shuffle_order_engine::max() const`** This should be constexpr.

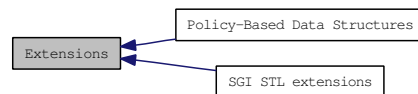
Member **`std::shuffle_order_engine::min() const`** This should be constexpr.

Chapter 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 STL extensions](#)
- [Policy-Based Data Structures](#)

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI STL extensions

Collaboration diagram for SGI STL extensions:



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, 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)
- 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, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `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 _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, 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)`
- `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)`

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 functions 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, typename _Compare> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c, _Compare __comp) [inline]`

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 571 of file `ext/algorithm`.

2.2.2.2 `template<typename _Tp> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c) [inline]`

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 537 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 1304 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 1315 of file bitset.

2.2.2.5 `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 963 of file `bitset`.

2.2.2.6 `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 145 of file `ext/functional`.

2.2.2.7 `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 172 of file `ext/functional`.

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

An [SGI extension](#) .

Definition at line 326 of file `ext/functional`.

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

An [SGI extension](#) .

Definition at line 332 of file `ext/functional`.

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

An [SGI extension](#) .

Definition at line 338 of file ext/functional.

```
2.2.2.11 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 119 of file ext/algorithm.

References `std::__iterator_category()`.

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

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 103 of file ext/iterator.

References `std::__iterator_category()`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::list<_Tp, _Alloc >::size()`.

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

An [SGI extension](#) .

Definition at line 93 of file ext/functional.

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

An [SGI extension](#) .

Definition at line 87 of file ext/functional.

2.2.2.15 `template<typename _ForwardIter , typename _Tp > void
__gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp
__value) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 132 of file ext/numeric.

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

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 453 of file ext/algorithm.

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

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 434 of file ext/algorithm.

```
2.2.2.18 template<typename _ForwardIterator , typename
           _StrictWeakOrdering > bool __gnu_cxx::is_sorted (_ForwardIterator
           __first, _ForwardIterator __last, _StrictWeakOrdering __comp)
           [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 503 of file ext/algorithm.

```
2.2.2.19 template<typename _ForwardIterator > bool __gnu_cxx::is_sorted
           (_ForwardIterator __first, _ForwardIterator __last) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 478 of file ext/algorithm.

```
2.2.2.20 template<typename _InputIterator1 , typename _InputIterator2 > int
           __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1,
           _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
           __last2) [inline]
```

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 200 of file ext/algorithm.

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

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 121 of file ext/numeric.

2.2.2.22 `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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 111 of file ext/numeric.

2.2.2.23 `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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 410 of file ext/algorithm.

```
2.2.2.24 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 387 of file ext/algorithm.

```
2.2.2.25 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) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 300 of file ext/algorithm.

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

```
2.2.2.26 template<typename _ForwardIterator , typename _OutputIterator ,
    typename _Distance > _OutputIterator _gnu_cxx::random_sample_n
    (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator
    __out, const _Distance __n) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 266 of file ext/algorithm.

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

```
2.2.2.27 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)`

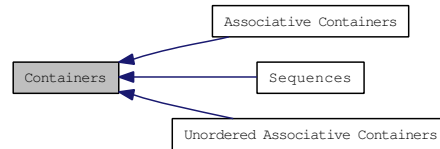
Like `copy()`, but does not require an initialized output range.

Definition at line 121 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 Containers](#)
- [Unordered Associative Containers](#)

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

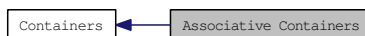
As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

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 Containers

Collaboration diagram for Associative Containers:



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

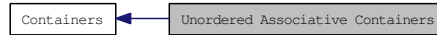
Associative containers allow fast retrieval of data based on keys.

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 Containers

Collaboration diagram for Unordered Associative Containers:



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

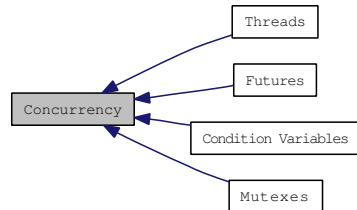
- [Exceptions](#)

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

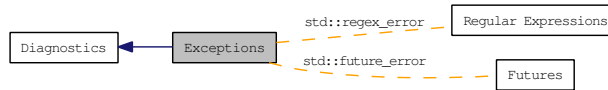
- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

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 &&,...) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&, const nested_exception *!=0) __-`
`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 ()`
- `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.html>

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 () [inline]

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

Definition at line 149 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 void std::rethrow_exception (exception_ptr)

Throw the object pointed to by the `exception_ptr`.

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

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

Overload, See N2619.

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

2.9.3.6 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.

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

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

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

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

2.9.3.9 `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.

2.9.3.10 `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.

2.9.3.11 `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()`.

2.9.3.12 `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

- `std::complex< double >::complex` (const `complex< long double > &`)
- `std::complex< float >::complex` (const `complex< long double > &`)
- `std::complex< float >::complex` (const `complex< double > &`)
- `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 >`
`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 >`
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `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 >`
`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 >`
`__gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp >`
`_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 >`
`__gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator*= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator*= (const _Tp &)`
- `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::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::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `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 _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp >`
`&)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
- `template<typename _Tp >`
`_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 >`
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const complex< _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 _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 _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 complex< _Tp > &__x, const`
`complex< _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 _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 _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 complex< _Tp > &__x, const complex<`
`_Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator== (const complex< _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 594 of file complex.

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

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

`acos(__z)` [8.1.2].

Definition at line 86 of file `tr1_impl/complex`.

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

`acosh(__z)` [8.1.5].

Definition at line 205 of file `tr1_impl/complex`.

2.11.2.4 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::arg (_Tp __x) [inline]`

Additional overloads [8.1.9].

Definition at line 311 of file `tr1_impl/complex`.

References `std::arg()`.

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

Return phase angle of z .

Definition at line 621 of file `complex`.

Referenced by `std::arg()`.

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

`asin(__z)` [8.1.3].

Definition at line 122 of file `tr1_impl/complex`.

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

`asinh(__z)` [8.1.6].

Definition at line 244 of file `tr1_impl/complex`.

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

`atan(__z)` [8.1.4].

Definition at line 166 of file `tr1_impl/complex`.

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

`atanh(__z)` [8.1.7].

Definition at line 288 of file `tr1_impl/complex`.

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

Return [complex](#) conjugate of `z`.

Definition at line 667 of file `complex`.

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

Return [complex](#) cosine of `z`.

Definition at line 699 of file `complex`.

Referenced by `std::polar()`.

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

Return [complex](#) hyperbolic cosine of `z`.

Definition at line 729 of file `complex`.

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

Return `complex` base e exponential of `z`.

Definition at line 755 of file `complex`.

Referenced by `std::pow()`.

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

`fabs(__z)` [8.1.8].

Definition at line 301 of file `tr1_impl/complex`.

References `std::abs()`.

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

Return `complex` natural logarithm of `z`.

Definition at line 782 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()`, `std::independent_bits_engine< _RandomNumberEngine, _w, _UIntType >::operator()`, and `std::pow()`.

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

Return `complex` base 10 logarithm of `z`.

Definition at line 787 of file `complex`.

References `std::log()`.

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

Return `z` magnitude squared.

Definition at line 654 of file `complex`.

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

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

Return false if x is equal to y .

Definition at line 479 of file `complex`.

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

Return false if x is equal to y .

Definition at line 474 of file `complex`.

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

Return false if x is equal to y .

Definition at line 469 of file `complex`.

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

Return false if x is equal to y .

Definition at line 397 of file `complex`.

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

Return false if x is equal to y .

Definition at line 388 of file `complex`.

2.11.2.23 `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 379 of file `complex`.

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

Multiply this `complex` number by `z`.

Definition at line 292 of file `complex`.

2.11.2.25 `template<typename _Tp > complex<_Tp > & std::complex<_Tp >::operator*=(const _Tp & _t) [inline, inherited]`

Multiply this `complex` number by `t`.

Definition at line 238 of file `complex`.

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

Return `x`.

Definition at line 438 of file `complex`.

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

Return false if `x` is equal to `y`.

Definition at line 337 of file `complex`.

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

Return false if `x` is equal to `y`.

Definition at line 328 of file `complex`.

2.11.2.29 `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 319 of file `complex`.

2.11.2.30 `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator+= (const complex<_Up > & _z) [inline, inherited]`

Add z to this `complex` number.

Definition at line 269 of file `complex`.

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

Return `complex` negation of x .

Definition at line 444 of file `complex`.

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

Return false if x is equal to y .

Definition at line 367 of file `complex`.

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

Return false if x is equal to y .

Definition at line 358 of file `complex`.

2.11.2.34 `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 349 of file `complex`.

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

Subtract z from this `complex` number.

Definition at line 280 of file `complex`.

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

Return false if x is equal to y .

Definition at line 427 of file `complex`.

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

Return false if x is equal to y .

Definition at line 418 of file `complex`.

2.11.2.38 `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 409 of file `complex`.

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

Divide this `complex` number by z .

Definition at line 305 of file `complex`.

References `std::norm()`.

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

Divide this `complex` number by t .

Definition at line 248 of file `complex`.

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

Insertion operator for `complex` values.

Definition at line 519 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()`.

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

Assign this [complex](#) number to [complex](#) z .

Definition at line 258 of file `complex`.

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

Assign this [complex](#) number to scalar t .

Definition at line 228 of file `complex`.

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

Return false if x is equal to y .

Definition at line 461 of file `complex`.

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

Return false if x is equal to y .

Definition at line 456 of file `complex`.

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

Return true if x is equal to y .

Definition at line 451 of file `complex`.

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

Extraction operator for `complex` values.

Definition at line 486 of file `complex`.

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

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

Return `complex` with magnitude *rho* and angle *theta*.

Definition at line 662 of file `complex`.

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

Referenced by `std::pow()`.

2.11.2.49 `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 1009 of file `complex`.

References `std::log()`, `std::polar()`, and `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 1003 of file `complex`.

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

Return *x* to the *y*'th power.

Definition at line 964 of file `complex`.

References `std::exp()`, `std::log()`, and `std::polar()`.

Referenced by `std::gamma_distribution<_RealType >::operator()()`, and `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 817 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 847 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 891 of file `complex`.

Referenced by `std::normal_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 918 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 946 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 54 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.
- 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 `launch` { `any`, `async`, `sync` }

Functions

- `std::__basic_future::__basic_future` (`future< _Res > &&`)
- `std::__basic_future::__basic_future` (`shared_future< _Res > &&`)
- `std::__basic_future::__basic_future` (`const shared_future< _Res > &`)
- static `_Setter< void, void >` `std::future_base::State::__setter` (`promise< void > *__prom`)
- `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`)
- `template<typename _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type >` `std::async` (`launch __-`
`policy, _Fn &&__fn, _Args &&...__args`)
- 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, typename... _ArgTypes>`
void `std::swap` (`packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y`)
- `template<typename _Res >`
void `std::swap` (`promise< _Res > &__x, promise< _Res > &__y`)

Variables

- `const error_category *const std::future_category`

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 59 of file `future`.

2.13.3 Variable Documentation

2.13.3.1 `const error_category* const std::future_category`

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

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_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_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_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_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_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`

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/bk01pt>

2.14.2 Typedef Documentation

2.14.2.1 typedef basic_filebuf<char> std::filebuf

One of the [I/O](#) .

Definition at line 137 of file `iosfwd`.

2.14.2.2 typedef basic_fstream<char> std::fstream

One of the [I/O](#) .

Definition at line 140 of file `iosfwd`.

2.14.2.3 typedef basic_ifstream<char> std::ifstream

One of the [I/O](#) .

Definition at line 138 of file `iosfwd`.

2.14.2.4 typedef basic_ios<char> std::ios

One of the [I/O](#) .

Definition at line 123 of file `iosfwd`.

2.14.2.5 typedef basic_iostream<char> std::iostream

One of the [I/O](#) .

Definition at line 132 of file `iosfwd`.

2.14.2.6 typedef basic_istream<char> std::istream

One of the [I/O](#) .

Definition at line 130 of file iosfwd.

2.14.2.7 typedef basic_istringstream<char> std::istringstream

One of the [I/O](#) .

Definition at line 134 of file iosfwd.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

One of the [I/O](#) .

Definition at line 139 of file iosfwd.

2.14.2.9 typedef basic_ostream<char> std::ostream

One of the [I/O](#) .

Definition at line 131 of file iosfwd.

2.14.2.10 typedef basic_ostringstream<char> std::ostringstream

One of the [I/O](#) .

Definition at line 135 of file iosfwd.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

One of the [I/O](#) .

Definition at line 129 of file iosfwd.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

One of the [I/O](#) .

Definition at line 133 of file iosfwd.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

One of the [I/O](#) .

Definition at line 136 of file iosfwd.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

One of the [I/O](#) .

Definition at line 152 of file iosfwd.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

One of the [I/O](#) .

Definition at line 155 of file iosfwd.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

One of the [I/O](#) .

Definition at line 153 of file iosfwd.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

One of the [I/O](#) .

Definition at line 143 of file iosfwd.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

One of the [I/O](#) .

Definition at line 147 of file iosfwd.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

One of the [I/O](#) .

Definition at line 145 of file iosfwd.

2.14.2.20 typedef basic_istream<wchar_t> std::wistream

One of the [I/O](#) .

Definition at line 149 of file iosfwd.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

One of the [I/O](#) .

Definition at line 154 of file iosfwd.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

One of the [I/O](#) .

Definition at line 146 of file iosfwd.

2.14.2.23 typedef basic_ostringstream<wchar_t> std::wostringstream

One of the [I/O](#) .

Definition at line 150 of file iosfwd.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

One of the [I/O](#) .

Definition at line 144 of file iosfwd.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

One of the [I/O](#) .

Definition at line 148 of file iosfwd.

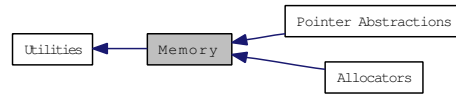
2.14.2.26 typedef basic_stringstream<wchar_t> std::wstringstream

One of the [I/O](#) .

Definition at line 151 of file iosfwd.

2.15 Memory

Collaboration diagram for Memory:



Modules

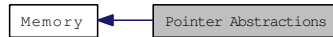
- [Pointer Abstractions](#)
- [Allocators](#)

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::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, _Tp_Deleter>`
20.7.12.2 `unique_ptr` for single objects.
- class `std::unique_ptr<_Tp[], _Tp_Deleter>`
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` (`_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 _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `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, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator== (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`

- `template<typename _Tp, typename _Tp_Deleter >`
`void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _Tp_Deleter > &__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 (_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 453 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 74 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 468 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 64 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 _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 &, _L2 &, _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**
- const adopt_lock_t **std::adopt_lock**
- const defer_lock_t **std::defer_lock**
- const try_to_lock_t **std::try_to_lock**

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) [**inline**]

call_once

Definition at line 722 of file mutex.

2.17.2.2 template<typename _L1 , typename _L2 , typename... _L3> void
std::lock (_L1 &, _L2 &, _L3 &...) [**inline**]

lock

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

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.

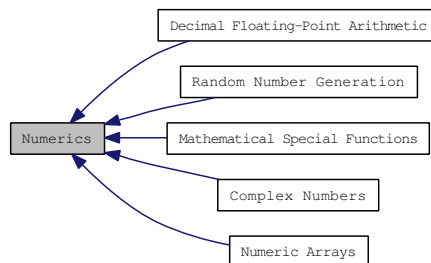
Definition at line 664 of file mutex.

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 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_greater< _R1, _R2 >`
ratio_greater
- struct `std::ratio_greater_equal< _R1, _R2 >`
ratio_greater_equal
- struct `std::ratio_less< _R1, _R2 >`
ratio_less
- struct `std::ratio_less_equal< _R1, _R2 >`
ratio_less_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< 1, 1000000000000000000 > std::atto`
- `typedef ratio< 1, 100 > std::centi`
- `typedef ratio< 10, 1 > std::deca`
- `typedef ratio< 1, 10 > std::deci`
- `typedef ratio< 1000000000000000000, 1 > std::exa`
- `typedef ratio< 1, 1000000000000000000 > std::femto`
- `typedef ratio< 1000000000, 1 > std::giga`
- `typedef ratio< 100, 1 > std::hecto`
- `typedef ratio< 1000, 1 > std::kilo`
- `typedef ratio< 1000000, 1 > std::mega`
- `typedef ratio< 1, 1000000 > std::micro`
- `typedef ratio< 1, 1000 > std::milli`
- `typedef ratio< 1, 1000000000 > std::nano`
- `typedef ratio< 1000000000000000000, 1 > std::peta`
- `typedef ratio< 1, 1000000000000 > std::pico`
- `typedef ratio< 1000000000000, 1 > std::tera`

Variables

- `static const intmax_t std::ratio::den`
- `static const intmax_t std::ratio::num`

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

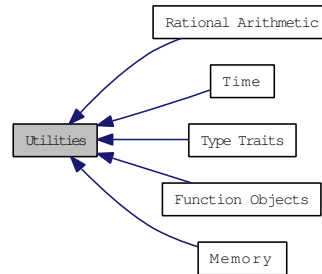
- bool `std::operator!=` (thread::id __x, thread::id __y)
- template<class _CharT, class _Traits >
basic_ostream< _CharT, _Traits > & `std::operator<<` (basic_ostream< _-
CharT, _Traits > &__out, thread::id __id)
- bool `std::operator<=` (thread::id __x, thread::id __y)
- bool `std::operator>` (thread::id __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](#)
- [Type Traits](#)
- [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](#) (const gslice &)
- [std::gslice::gslice](#) (size_t, const valarray< size_t > &, const valarray< size_t > &)
- [std::gslice::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](#) (size_t, size_t, size_t)
- [std::slice::slice](#) ()
- [std::slice_array::slice_array](#) (const slice_array &)
- `template<class _Dom >`
[std::valarray::valarray](#) (const _Expr< _Dom, _Tp > &_e)
- [std::valarray::valarray](#) (initializer_list< _Tp >)
- [std::valarray::valarray](#) (const indirect_array< _Tp > &)
- [std::valarray::valarray](#) (const mask_array< _Tp > &)
- [std::valarray::valarray](#) (const gslice_array< _Tp > &)
- [std::valarray::valarray](#) (const slice_array< _Tp > &)
- [std::valarray::valarray](#) (const valarray &)
- `template<typename _Tp>`
[std::valarray::valarray](#) (const _Tp *__restrict__ __p, size_t __n)
- [std::valarray::valarray](#) (const _Tp &, size_t)
- [std::valarray::valarray](#) (size_t)
- [std::valarray::valarray](#) ()
- [std::gslice::~gslice](#) ()
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp >` [std::valarray::apply](#) (_Tp func(const _Tp &)) const
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp >` [std::valarray::apply](#) (_Tp func(_Tp)) const
- `valarray< _Tp >` [std::valarray::cshift](#) (int) const
- `_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, _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, _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, _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, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &_t,`
`const valarray< _Tp > &_v)`
- `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, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _-`
`Tp > &_v, const valarray< _Tp > &_w)`
- `template<class _Dom >`
`void std::slice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::mask_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator%= (const _Expr< _Dom, _Tp >`
`&)`
- `valarray< _Tp > & std::valarray::operator%= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator%= (const _Tp &)`
- `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, _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, _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 >`
`void std::slice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `void std::mask_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator&= (const _Expr< _Dom, _Tp >`
`&)`
- `valarray< _Tp > & std::valarray::operator&= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator&= (const _Tp &)`
- `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, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _-`
`_Tp > &_v, const _Tp &_t)`
- `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<class _Dom >`
`void std::slice_array::operator* = (const _Expr< _Dom, _Tp > &) const`

- void `std::slice_array::operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::mask_array::operator*=(const _Expr<_Dom, _Tp> &) const`
- void `std::mask_array::operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::indirect_array::operator*=(const _Expr<_Dom, _Tp> &) const`
- void `std::indirect_array::operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::gslice_array::operator*=(const _Expr<_Dom, _Tp> &) const`
- void `std::gslice_array::operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`
valarray<_Tp> & `std::valarray::operator*=(const _Expr<_Dom, _Tp> &)`
- valarray<_Tp> & `std::valarray::operator*=(const valarray<_Tp> &)`
- valarray<_Tp> & `std::valarray::operator*=(const _Tp &)`
- `_UnaryOp<__unary_plus>::_Rt std::valarray::operator+()` const
- `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<__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<class _Dom>`
void `std::slice_array::operator+=(const _Expr<_Dom, _Tp> &) const`
- void `std::slice_array::operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::mask_array::operator+=(const _Expr<_Dom, _Tp> &) const`
- void `std::mask_array::operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::indirect_array::operator+=(const _Expr<_Dom, _Tp> &) const`
- void `std::indirect_array::operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`
void `std::gslice_array::operator+=(const _Expr<_Dom, _Tp> &) const`
- void `std::gslice_array::operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`
valarray<_Tp> & `std::valarray::operator+=(const _Expr<_Dom, _Tp>`
`&)`
- valarray<_Tp> & `std::valarray::operator+=(const valarray<_Tp> &)`

- `valarray< _Tp > & std::valarray::operator+= (const _Tp &)`
- `_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<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, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&_v, const valarray< _Tp > &_w)`
- `template<class _Dom >`
`void std::slice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array::operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::mask_array::operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator-= (const _Expr< _Dom, _Tp >`
`&)`
- `valarray< _Tp > & std::valarray::operator-= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator-= (const _Tp &)`
- `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< __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, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&_v, const valarray< _Tp > &_w)`
- `template<class _Dom >`
`void std::slice_array::operator/= (const _Expr< _Dom, _Tp > &) const`

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

- void `std::mask_array::operator<<=` (const valarray< _Tp > &) const
- template<class _Dom >
void `std::indirect_array::operator<<=` (const _Expr< _Dom, _Tp > &) const

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

- `gslice_array & std::gslice_array::operator=` (const `gslice_array` &)
- `gslice & std::gslice::operator=` (const `gslice` &)
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator=` (const `_Expr< _Dom, _Tp >` &)

- `valarray< _Tp > & std::valarray::operator=` (const `indirect_array< _Tp >` &)
- `valarray< _Tp > & std::valarray::operator=` (const `mask_array< _Tp >` &)
- `valarray< _Tp > & std::valarray::operator=` (const `gslice_array< _Tp >` &)
- `valarray< _Tp > & std::valarray::operator=` (const `slice_array< _Tp >` &)
- `valarray< _Tp > & std::valarray::operator=` (const `_Tp` &)
- `valarray & std::valarray::operator=` (`initializer_list< _Tp >`)
- `valarray< _Tp > & std::valarray::operator=` (const `valarray< _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, _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, _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_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< __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< __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<class _Dom >`
`void std::slice_array::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::mask_array::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator>>= (const _Expr< _Dom, _Tp`
`> &)`
- `valarray< _Tp > & std::valarray::operator>>= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator>>= (const _Tp &)`
- `indirect_array< _Tp > std::valarray::operator[] (const valarray< size_t > &)`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`
`valarray< size_t > &) const`
- `mask_array< _Tp > std::valarray::operator[] (const valarray< bool > &)`
- `valarray< _Tp > std::valarray::operator[] (const valarray< bool > &) const`
- `gslice_array< _Tp > std::valarray::operator[] (const gslice &)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`
`gslice &) const`
- `slice_array< _Tp > std::valarray::operator[] (slice)`

- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (slice) const
- `_Tp & std::valarray::operator[]` (size_t)
- `const _Tp & std::valarray::operator[]` (size_t) const
- `template<typename _Tp > _Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const _Tp &__t, const valarray< _Tp > &__v)
- `template<typename _Tp > _Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const valarray< _Tp > &__v, const _Tp &__t)
- `template<typename _Tp > _Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- `template<class _Dom > void std::slice_array::operator^=` (const _Expr< _Dom, _Tp > &) const
- `void std::slice_array::operator^=` (const valarray< _Tp > &) const
- `template<class _Dom > void std::mask_array::operator^=` (const _Expr< _Dom, _Tp > &) const
- `void std::mask_array::operator^=` (const valarray< _Tp > &) const
- `template<class _Dom > void std::indirect_array::operator^=` (const _Expr< _Dom, _Tp > &) const
- `void std::indirect_array::operator^=` (const valarray< _Tp > &) const
- `template<class _Dom > void std::gslice_array::operator^=` (const _Expr< _Dom, _Tp > &) const
- `void std::gslice_array::operator^=` (const valarray< _Tp > &) const
- `template<class _Dom > valarray< _Tp > & std::valarray::operator^=` (const _Expr< _Dom, _Tp > & &)
- `valarray< _Tp > & std::valarray::operator^=` (const valarray< _Tp > &)
- `valarray< _Tp > & std::valarray::operator^=` (const _Tp &)
- `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)
- `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<class _Dom >`
`void std::slice_array::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::mask_array::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator|= (const _Expr< _Dom, _Tp >`
`&)`
- `valarray< _Tp > & std::valarray::operator|= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator|= (const _Tp &)`
- `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)`
- `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)`
- `_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::slice::start () const`
- `size_t std::gslice::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 (const gslice & __g) [inline, inherited]`

Copy constructor.

Definition at line 156 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 151 of file `gslice.h`.

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

Construct an empty [slice](#).

Definition at line 147 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 142 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 142 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 138 of file `mask_array.h`.

2.22.2.7 `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 93 of file `slice_array.h`.

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

Construct an empty [slice](#).

Definition at line 89 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 206 of file `slice_array.h`.

2.22.2.10 `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 649 of file `valarray`.

2.22.2.11 `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 639 of file `valarray`.

2.22.2.12 `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 630 of file valarray.

2.22.2.13 `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 619 of file valarray.

2.22.2.14 `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 610 of file valarray.

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

Copy constructor.

Definition at line 603 of file valarray.

2.22.2.16 `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 588 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 582 of file valarray.

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

Construct an empty [array](#).

Definition at line 578 of file `valarray`.

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

Destructor.

Definition at line 161 of file `gslice.h`.

2.22.2.20 `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 980 of file `valarray`.

2.22.2.21 `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 972 of file `valarray`.

2.22.2.22 `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 898 of file `valarray`.

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

Return the maximum element using operator<().

Definition at line 964 of file `valarray`.

References `std::max_element()`.

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

Return the minimum element using operator<().

Definition at line 956 of file `valarray`.

References `std::min_element()`.

2.22.2.25 `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 999 of file `valarray`.

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

Modulo [slice](#) elements by corresponding elements of *v*.

Definition at line 257 of file `slice_array.h`.

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

Modulo [slice](#) elements by corresponding elements of *v*.

Definition at line 191 of file `mask_array.h`.

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

Modulo [slice](#) elements by corresponding elements of *v*.

Definition at line 195 of file `indirect_array.h`.

2.22.2.29 `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 201 of file `gslice_array.h`.

2.22.2.30 `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 1026 of file `valarray`.

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 1026 of file valarray.

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

Logical and [slice](#) elements with corresponding elements of *v*.

Definition at line 261 of file slice_array.h.

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

Logical and [slice](#) elements with corresponding elements of *v*.

Definition at line 195 of file mask_array.h.

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

Logical and [slice](#) elements with corresponding elements of *v*.

Definition at line 199 of file indirect_array.h.

2.22.2.35 `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 205 of file gslice_array.h.

2.22.2.36 `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 1028 of file valarray.

2.22.2.37 `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 1028 of file `valarray`.

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

Multiply `slice` elements by corresponding elements of `v`.

Definition at line 255 of file `slice_array.h`.

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

Multiply `slice` elements by corresponding elements of `v`.

Definition at line 189 of file `mask_array.h`.

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

Multiply `slice` elements by corresponding elements of `v`.

Definition at line 193 of file `indirect_array.h`.

2.22.2.41 `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 199 of file `gslice_array.h`.

2.22.2.42 `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 1024 of file `valarray`.

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

Multiply each element of `array` by `t`.

Definition at line 1024 of file `valarray`.

2.22.2.44 `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 996 of file `valarray`.

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

Add corresponding elements of `v` to `slice` elements.

Definition at line 258 of file `slice_array.h`.

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

Add corresponding elements of `v` to `slice` elements.

Definition at line 192 of file `mask_array.h`.

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

Add corresponding elements of `v` to `slice` elements.

Definition at line 196 of file `indirect_array.h`.

2.22.2.48 `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 202 of file `gslice_array.h`.

2.22.2.49 `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 1022 of file `valarray`.

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

Add `t` to each element of `array`.

Definition at line 1022 of file `valarray`.

2.22.2.51 `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 997 of file `valarray`.

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

Subtract corresponding elements of `v` from `slice` elements.

Definition at line 259 of file `slice_array.h`.

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

Subtract corresponding elements of `v` from `slice` elements.

Definition at line 193 of file `mask_array.h`.

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

Subtract corresponding elements of *v* from [slice](#) elements.

Definition at line 197 of file `indirect_array.h`.

2.22.2.55 `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 203 of file `gslice_array.h`.

2.22.2.56 `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 1023 of file `valarray`.

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

Subtract *t* to each element of [array](#).

Definition at line 1023 of file `valarray`.

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

Divide [slice](#) elements by corresponding elements of *v*.

Definition at line 256 of file `slice_array.h`.

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

Divide [slice](#) elements by corresponding elements of *v*.

Definition at line 190 of file `mask_array.h`.

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

Divide [slice](#) elements by corresponding elements of *v*.

Definition at line 194 of file indirect_array.h.

2.22.2.61 `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 200 of file gslice_array.h.

2.22.2.62 `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 1025 of file valarray.

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

Divide each element of [array](#) by *t*.

Definition at line 1025 of file valarray.

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

Left shift [slice](#) elements by corresponding elements of *v*.

Definition at line 263 of file slice_array.h.

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

Left shift [slice](#) elements by corresponding elements of *v*.

Definition at line 197 of file mask_array.h.

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

Left shift [slice](#) elements by corresponding elements of *v*.

Definition at line 201 of file indirect_array.h.

2.22.2.67 `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 207 of file gslice_array.h.

2.22.2.68 `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 1030 of file valarray.

2.22.2.69 `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 1030 of file valarray.

2.22.2.70 `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 228 of file slice_array.h.

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

Assign all [slice](#) elements to *t*.

Definition at line 223 of file slice_array.h.

2.22.2.72 `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 214 of file slice_array.h.

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

Assign all [slice](#) elements to *t*.

Definition at line 157 of file mask_array.h.

2.22.2.74 `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 148 of file mask_array.h.

2.22.2.75 `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 167 of file indirect_array.h.

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

Assign all [slice](#) elements to *t*.

Definition at line 162 of file indirect_array.h.

2.22.2.77 `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 153 of file indirect_array.h.

2.22.2.78 `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 165 of file gslice_array.h.

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

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

Assign all [slice](#) elements to *t*.

Definition at line 157 of file gslice_array.h.

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

2.22.2.80 `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 147 of file gslice_array.h.

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

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

Assignment operator.

Definition at line 168 of file gslice.h.

2.22.2.82 `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 756 of file `valarray`.

2.22.2.83 `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 746 of file `valarray`.

2.22.2.84 `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 736 of file `valarray`.

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

2.22.2.85 `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 726 of file valarray.

2.22.2.86 `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 718 of file valarray.

2.22.2.87 `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 694 of file valarray.

2.22.2.88 `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 670 of file valarray.

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

Right shift [slice](#) elements by corresponding elements of *v*.

Definition at line 264 of file `slice_array.h`.

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

Right shift [slice](#) elements by corresponding elements of *v*.

Definition at line 198 of file `mask_array.h`.

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

Right shift [slice](#) elements by corresponding elements of *v*.

Definition at line 202 of file `indirect_array.h`.

2.22.2.92 `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 208 of file `gslice_array.h`.

2.22.2.93 `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 1031 of file `valarray`.

2.22.2.94 `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 1031 of file valarray.

2.22.2.95 `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 836 of file valarray.

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

2.22.2.96 `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 828 of file valarray.

2.22.2.97 `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 817 of file `valarray`.

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

2.22.2.98 `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 805 of file `valarray`.

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

2.22.2.99 `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 797 of file `valarray`.

```
2.22.2.100 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 788 of file `valarray`.

```
2.22.2.101 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 783 of file `valarray`.

2.22.2.102 `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 775 of file `valarray`.

2.22.2.103 `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 552 of file `valarray`.

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

Logical xor [slice](#) elements with corresponding elements of *v*.

Definition at line 260 of file `slice_array.h`.

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

Logical xor [slice](#) elements with corresponding elements of *v*.

Definition at line 194 of file `mask_array.h`.

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

Logical xor [slice](#) elements with corresponding elements of *v*.

Definition at line 198 of file `indirect_array.h`.

2.22.2.107 `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 204 of file `gslice_array.h`.

2.22.2.108 `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 1027 of file `valarray`.

2.22.2.109 `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 1027 of file `valarray`.

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

Logical or [slice](#) elements with corresponding elements of *v*.

Definition at line 262 of file slice_array.h.

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

Logical or [slice](#) elements with corresponding elements of *v*.

Definition at line 196 of file mask_array.h.

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

Logical or [slice](#) elements with corresponding elements of *v*.

Definition at line 200 of file indirect_array.h.

2.22.2.113 `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 206 of file gslice_array.h.

2.22.2.114 `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 1029 of file valarray.

2.22.2.115 `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 1029 of file valarray.

2.22.2.116 `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 998 of file `valarray`.

2.22.2.117 `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 939 of file `valarray`.

2.22.2.118 `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 857 of file `valarray`.

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

Return size of [slice](#).

Definition at line 101 of file `slice_array.h`.

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

Return [array](#) of sizes of [slice](#) dimensions.

Definition at line 137 of file `gslice.h`.

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

Return the number of elements in [array](#).

Definition at line 844 of file `valarray`.

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

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

Return [array](#) offset of first [slice](#) element.

Definition at line 97 of file `slice_array.h`.

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

Return [array](#) offset of first [slice](#) element.

Definition at line 133 of file `gslice.h`.

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

Return [array](#) stride of [slice](#).

Definition at line 105 of file `slice_array.h`.

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

Return [array](#) of [array](#) strides for each dimension.

Definition at line 141 of file `gslice.h`.

2.22.2.126 `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 849 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 123 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 140 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 157 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 174 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 191 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 208 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 225 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 242 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 259 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 276 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 293 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 310 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 327 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 344 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 361 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 378 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 395 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 412 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 429 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 446 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 463 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 480 of file tr1/cmath.

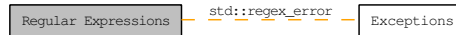
2.23.2.23 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_neumann (unsigned int __n, _Tp __x) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 497 of file tr1/cmath.

2.24 Regular Expressions

Collaboration diagram for 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_error](#)
*A regular expression **exception** class.
The regular expression library throws objects of this class on error.*
- 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< _BIter >](#)

Namespaces

- namespace [std::regex_constants](#)

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::wregex_iterator**
- typedef regex_token_iterator< const wchar_t * > **std::wregex_token_iterator**
- typedef sub_match< const wchar_t * > **std::wsub_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

- bool **std::regex_traits::isctype** (_Ch_type __c, char_class_type __f) const
- 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 >
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 _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 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 _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< (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, 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, 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 _BiIter >`
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__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 >`
`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 _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 _BiIter >`
`bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__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== (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 _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 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 _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> (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, 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, 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)`

- ```
>::value_type, _Ch_traits, _Ch_alloc > &__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 >`  
`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, 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, 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 _BiIter >`  
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__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 _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 _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 _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, 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 _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 _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_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 _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)`
- `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 _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 _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_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_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 _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 _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)`

### 2.24.1 Detailed Description

A facility for performing regular expression pattern matching.

### 2.24.2 Typedef Documentation

#### 2.24.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token [iterator](#) for C-style NULL-terminated strings.

Definition at line 2697 of file `tr1_impl/regex`.

#### 2.24.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 1178 of file `tr1_impl/regex`.

#### 2.24.2.3 `typedef basic_regex<char> std::regex`

Standard regular expressions.

Definition at line 1054 of file `tr1_impl/regex`.

**2.24.2.4 typedef regex\_token\_iterator<string::const\_iterator>  
std::sregex\_token\_iterator**

Token [iterator](#) for standard strings.

Definition at line 2699 of file tr1\_impl/regex.

**2.24.2.5 typedef sub\_match<string::const\_iterator> std::ssub\_match**

Standard regex submatch over a standard string.

Definition at line 1180 of file tr1\_impl/regex.

**2.24.2.6 typedef regex\_token\_iterator<const wchar\_t\*>  
std::wcregex\_token\_iterator**

Token [iterator](#) for C-style NULL-terminated wide strings.

Definition at line 2702 of file tr1\_impl/regex.

**2.24.2.7 typedef sub\_match<const wchar\_t\*> std::wsub\_match**

Regex submatch over a C-style null-terminated wide string.

Definition at line 1183 of file tr1\_impl/regex.

**2.24.2.8 typedef basic\_regex<wchar\_t> std::wregex**

Standard wide-character regular expressions.

Definition at line 1057 of file tr1\_impl/regex.

**2.24.2.9 typedef regex\_token\_iterator<wstring::const\_iterator>  
std::wsregex\_token\_iterator**

Token [iterator](#) for standard wide-character strings.

Definition at line 2704 of file tr1\_impl/regex.

**2.24.2.10 typedef sub\_match<wstring::const\_iterator> std::wssub\_match**

Regex submatch over a standard wide string.

Definition at line 1185 of file tr1\_impl/regex.

### 2.24.3 Function Documentation

**2.24.3.1** `template<typename _Ch_type > bool std::regex_traits<_Ch_type >::isctype (_Ch_type __c, char_class_type __f) const [inline, 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 655 of file `tr1_impl/regex`.

References `std::__ctype_abstract_base<_CharT >::is()`, `std::regex_traits<_Ch_type >::lookup_classname()`, `std::use_facet()`, and `std::__ctype_abstract_base<_CharT >::widen()`.

**2.24.3.2** `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 2072 of file `tr1_impl/regex`.

**2.24.3.3** `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 1673 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.4** `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 1599 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.5** `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 1525 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.6** `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 1451 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.7** `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 1367 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.8** `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 1286 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.9** `template<typename _BiIter > bool std::operator!=(const sub_match<_BiIter > & __lhs, const sub_match<_BiIter > & __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 1210 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::compare()`.

**2.24.3.10** `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 1685 of file `tr1_impl/regex`.

**2.24.3.11** `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 1611 of file tr1\_impl/regex.

**2.24.3.12** `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 1537 of file tr1\_impl/regex.

**2.24.3.13** `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 1463 of file `tr1_impl/regex`.

**2.24.3.14** `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 1381 of file `tr1_impl/regex`.

**2.24.3.15** `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 1299 of file `tr1_impl/regex`.

References `std::sub_match< _BiIter >::str()`.

**2.24.3.16** `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 1222 of file tr1\_impl/regex.

**2.24.3.17** `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 1736 of file tr1\_impl/regex.

**2.24.3.18** `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 1721 of file `tr1_impl/regex`.

**2.24.3.19** `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 1647 of file `tr1_impl/regex`.

**2.24.3.20** `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 1573 of file `tr1_impl/regex`.

**2.24.3.21** `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 1499 of file tr1\_impl/regex.

**2.24.3.22** `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 1423 of file tr1\_impl/regex.

**2.24.3.23** `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 1338 of file tr1\_impl/regex.

References `std::sub_match< _BiIter >::str()`.

```
2.24.3.24 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 1234 of file tr1\_impl/regex.

```
2.24.3.25 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.

```
2.24.3.26 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 1660 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.27** `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 1586 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.28** `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 1512 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.29** `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 1438 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.30** `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 1352 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.31** `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 1271 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.32** `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 1198 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::compare()`.

**2.24.3.33** `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 1697 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.34** `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 1623 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.35** `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 string.

**Returns:**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1549 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.36** `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 1475 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.37** `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 1395 of file tr1\_impl/regex.

References `std::sub_match< _BiIter >::str()`.

**2.24.3.38** `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 1312 of file tr1\_impl/regex.

References `std::sub_match< _BiIter >::str()`.

**2.24.3.39** `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 1258 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::compare()`.

**2.24.3.40** `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 1709 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.41** `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 1635 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter >::str()`.

**2.24.3.42** `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 string.

**Returns:**

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1561 of file `tr1_impl/regex`.

References `std::sub_match< _BiIter >::str()`.

**2.24.3.43** `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 1487 of file `tr1_impl/regex`.

References `std::sub_match< _BiIter >::str()`.

**2.24.3.44** `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 1409 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.45** `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 1325 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::str()`.

**2.24.3.46** `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 1246 of file `tr1_impl/regex`.

References `std::sub_match<_BiIter>::compare()`.

```

2.24.3.47 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 2232 of file `tr1_impl/regex`.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```

2.24.3.48 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 2210 of file `tr1_impl/regex`.

References `std::regex_match()`.

```
2.24.3.49 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 2187 of file `tr1_impl/regex`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```

2.24.3.50 template<typename _Ch_type , typename _Allocator ,
typename _Rx_traits > bool std::regex_match (const
_Ch_type * __s, match_results< const _Ch_type *,
_Alocator > & __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 2163 of file `tr1_impl/regex`.

References `std::regex_match()`.

```

2.24.3.51 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) [inline]

```

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 2138 of file `tr1_impl/regex`.

References `std::regex_match()`.

```
2.24.3.52 template<typename _Bi_iter , typename _Allocator , typename
_Ch_type , typename _Rx_traits > bool std::regex_match
(_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]
```

Determines 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.

*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.

Referenced by `std::regex_match()`.

**2.24.3.53** `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**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**Parameters:**

*s*  
*e*  
*fmt*  
*flags*

**Returns:**

a copy of string *s* with replacements.

**Exceptions:**

*an* exception of type `regex_error`.

Definition at line 2409 of file `tr1_impl/regex`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

**2.24.3.54** `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**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**Parameters:**

*out*

*first*  
*last*  
*e*  
*fmt*  
*flags*

**Returns:**

out

**Exceptions:**

*an* [exception](#) of type `regex_error`.

**Todo**

Implement this function.

Referenced by `std::regex_replace()`.

```
2.24.3.55 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 2364 of file tr1\_impl/regex.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

```
2.24.3.56 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**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**Exceptions:**

- an* exception of type `regex_error`.

Definition at line 2341 of file tr1\_impl/regex.

References `std::regex_search()`.

```
2.24.3.57 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**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Exceptions:**

*an* exception of type `regex_error`.

Definition at line 2321 of file `tr1_impl/regex`.

References `std::regex_search()`.

```
2.24.3.58 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**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Exceptions:**

*an* exception of type `regex_error`.

Definition at line 2301 of file `tr1_impl/regex`.

References `std::regex_search()`.

```
2.24.3.59 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**

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

**Exceptions:**

*an* exception of type `regex_error`.

Definition at line 2277 of file `tr1_impl/regex`.

References `std::regex_search()`.

```
2.24.3.60 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.

Referenced by `std::regex_search()`.

```
2.24.3.61 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 2086 of file `tr1_impl/regex`.

References `std::match_results< _Bi_iter, _Allocator >::swap()`.

```
2.24.3.62 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 1069 of file tr1\_impl/regex.

References `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

**2.24.3.63** `template<typename _Ch_type > int std::regex_traits<_Ch_type >::value(_Ch_type __ch, int __radix) const [inline, 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 690 of file tr1\_impl/regex.

References `std::basic_ios<_CharT, _Traits >::fail()`, `std::hex()`, and `std::oct()`.



## 2.25 Type Traits

Collaboration diagram for Type Traits:



### Classes

- 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_assign< _Tp >`  
*has\_nothrow\_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_assign< _Tp >`  
*has\_trivial\_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_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_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`

## 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 ()`

## Variables

- `static const _Tp std::integral\_constant::value`

### 2.25.1 Detailed Description

Compile time type transformation and information.

### 2.25.2 Typedef Documentation

#### 2.25.2.1 `typedef integral_constant<bool, false> std::false\_type`

typedef for `false_type`

Definition at line 78 of file `tr1_impl/type_traits`.

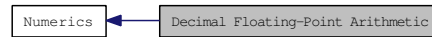
#### 2.25.2.2 `typedef integral_constant<bool, true> std::true\_type`

typedef for `true_type`

Definition at line 75 of file `tr1_impl/type_traits`.

## 2.26 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



### Namespaces

- namespace [std::decimal](#)

#### 2.26.1 Detailed Description

Classes and functions for [decimal](#) floating-point arithmetic.

## 2.27 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 `_Func` , typename... `_ArgTypes`>  
`_Bind< typename _Maybe_wrap_member_pointer< _Func >::type(_-  
ArgTypes...)> std::bind ( _Func __f, _ArgTypes... __args)`



- `template<typename _Operation , typename _Tp >`  
`binder1st< _Operation > std::binder1st (const _Operation &__fn, const _Tp &_`  
`_x)`
- `template<typename _Operation , typename _Tp >`  
`binder2nd< _Operation > std::binder2nd (const _Operation &__fn, const _Tp &_`  
`_x)`

### 2.27.1 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 `binder1st` 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 `binder2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `binder2nd(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 `binder1st` had been used, the functor would perform `1.3 - x` instead.)

Creator-wrapper functions like `binder1st` 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 `binder1st(std::plus<int>, 5)`.)

These become more useful when combined with the composition functions.

### 2.27.2 Function Documentation

**2.27.2.1** `template<typename _Functor , typename... _ArgTypes>`  
`_Bind<typename _Maybe_wrap_member_pointer<_-`  
`Functor>::type(_ArgTypes...)> std::bind (_Functor _f, _ArgTypes...`  
`_args) [inline]`

Function template for `std::bind`.

Definition at line 1372 of file `functional`.

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

One of the [binder functors](#).

Definition at line 125 of file binders.h.

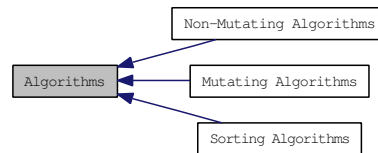
**2.27.2.3** `template<typename _Operation , typename _Tp >  
binder2nd<_Operation> std::bind2nd (const _Operation & __fn,  
const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 160 of file binders.h.

## 2.28 Algorithms

Collaboration diagram for Algorithms:



### Modules

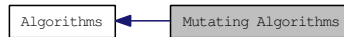
- [Mutating Algorithms](#)
- [Non-Mutating Algorithms](#)
- [Sorting Algorithms](#)

#### 2.28.1 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.

## 2.29 Mutating Algorithms

Collaboration diagram for Mutating Algorithms:



### 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, 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 _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 _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, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result)`

## 2.29.1 Function Documentation

### 2.29.1.1 `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 464 of file `stl_algobase.h`.

```
2.29.1.2 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 633 of file `stl_algobase.h`.

```
2.29.1.3 template<typename _InputIterator , typename _OutputIterator ,
typename _Predicate > _OutputIterator std::copy_if (_InputIterator
__first, _InputIterator __last, _OutputIterator __result, _Predicate
__pred) [inline]
```

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 945 of file `stl_algo.h`.

**2.29.1.4** `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 1002 of file `stl_algo.h`.

References `std::__iterator_category()`.

**2.29.1.5** `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 735 of file `stl_algobase.h`.

```
2.29.1.6 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 793 of file `stl_algobase.h`.

```
2.29.1.7 template<typename _ForwardIterator , typename _Generator > void
std::generate (_ForwardIterator __first, _ForwardIterator __last,
_Generator __gen) [inline]
```

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 4832 of file `stl_algo.h`.

**2.29.1.8** `template<typename _OutputIterator , typename _Size , typename _Generator > _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen) [inline]`

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 4863 of file `stl_algo.h`.

**2.29.1.9** `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 797 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

**2.29.1.10** `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 117 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()`, and `std::swap_ranges()`.

**2.29.1.11** `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 497 of file `stl_algobase.h`.

**2.29.1.12** `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 81 of file `move.h`.

**2.29.1.13** `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 669 of file stl\_algobase.h.

```
2.29.1.14 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 5030 of file stl\_algo.h.

References `std::__iterator_category()`, and `std::__partition()`.

```
2.29.1.15 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)
[inline]
```

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 1031 of file `stl_algo.h`.

**2.29.1.16** `template<typename _ForwardIterator , typename _Predicate >  
_ForwardIterator std::partition_point (_ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred) [inline]`

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 815 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

**2.29.1.17** `template<typename _RandomAccessIterator , typename  
_RandomNumberGenerator > void std::random_shuffle  
(_RandomAccessIterator __first, _RandomAccessIterator __last,  
_RandomNumberGenerator & __rand) [inline]`

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 4998 of file `stl_algo.h`.

References `std::iter_swap()`.

```
2.29.1.18 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 4970 of file `stl_algo.h`.

References `std::iter_swap()`.

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

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 1080 of file `stl_algo.h`.

**2.29.1.20** `template<typename _InputIterator , typename _OutputIterator ,  
typename _Tp > _OutputIterator std::remove_copy (_InputIterator  
__first, _InputIterator __last, _OutputIterator __result, const _Tp &  
__value) [inline]`

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 868 of file `stl_algo.h`.

**2.29.1.21** `template<typename _InputIterator , typename _OutputIterator ,  
typename _Predicate > _OutputIterator std::remove_copy_if  
(_InputIterator __first, _InputIterator __last, _OutputIterator  
__result, _Predicate __pred) [inline]`

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 906 of file `stl_algo.h`.

```
2.29.1.22 template<typename _ForwardIterator , typename _Predicate
> _ForwardIterator std::remove_if (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred) [inline]
```

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 1123 of file `stl_algo.h`.

```
2.29.1.23 template<typename _ForwardIterator , typename _Tp > void
std::replace (_ForwardIterator __first, _ForwardIterator __last,
const _Tp & __old_value, const _Tp & __new_value) [inline]
```

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 4768 of file `stl_algo.h`.

**2.29.1.24** `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) [inline]`

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 3850 of file `stl_algo.h`.

**2.29.1.25** `template<typename _ForwardIterator , typename _Predicate  
, typename _Tp > void std::replace_if (_ForwardIterator __first,  
_FowardIterator __last, _Predicate __pred, const _Tp &  
__new_value) [inline]`

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 4800 of file `stl_algo.h`.

```
2.29.1.26 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 1431 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

```
2.29.1.27 template<typename _BidirectionalIterator , typename
_OutputIterator > _OutputIterator std::reverse_copy
(_BidirectionalIterator __first, _BidirectionalIterator __last,
_OutputIterator __result) [inline]
```

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 1458 of file `stl_algo.h`.

**2.29.1.28** `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 1662 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()`.

```
2.29.1.29 template<typename _ForwardIterator , typename _OutputIterator
> _OutputIterator std::rotate_copy (_ForwardIterator
 __first, _ForwardIterator __middle, _ForwardIterator __last,
 _OutputIterator __result) [inline]
```

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 1696 of file stl\_algo.h.

```
2.29.1.30 template<typename _ForwardIterator , typename _Predicate >
 _ForwardIterator std::stable_partition (_ForwardIterator __first,
 _ForwardIterator __last, _Predicate __pred) [inline]
```

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 1854 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()`.

### 2.29.1.31 `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 106 of file `move.h`.

### 2.29.1.32 `template<typename _ForwardIterator1 , typename _ForwardIterator2 > _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2) [inline]`

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 158 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

**2.29.1.33** `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _BinaryOperation > _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op) [inline]`

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 4736 of file `stl_algo.h`.

**2.29.1.34** `template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation > _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op) [inline]`

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 4700 of file `stl_algo.h`.

**2.29.1.35** `template<typename _ForwardIterator, typename _BinaryPredicate  
> _ForwardIterator std::unique (_ForwardIterator __first,  
_ForwardIterator __last, _BinaryPredicate __binary_pred)  
[inline]`

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 1203 of file `stl_algo.h`.

**2.29.1.36** `template<typename _ForwardIterator > _ForwardIterator  
std::unique (_ForwardIterator __first, _ForwardIterator __last)  
[inline]`

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 1163 of file `stl_algo.h`.

```
2.29.1.37 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 4939 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

**2.29.1.38** `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 CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

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

References `std::__iterator_category()`, and `std::__unique_copy()`.

## 2.30 Non-Mutating Algorithms

Collaboration diagram for Non-Mutating Algorithms:



### Functions

- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `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 _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`bool std::equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _`  
`BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `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, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __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 _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 _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 _InputIterator, typename _Predicate >`  
`bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __-`  
`pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`  
`__predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `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)`

### 2.30.1 Function Documentation

**2.30.1.1** `template<typename _ForwardIterator, typename _BinaryPredicate  
> _ForwardIterator std::adjacent_find (_ForwardIterator __first,  
_ForwardIterator __last, _BinaryPredicate __binary_pred)  
[inline]`

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 4379 of file `stl_algo.h`.

**2.30.1.2** `template<typename _ForwardIterator > _ForwardIterator  
std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)  
[inline]`

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 4347 of file `stl_algo.h`.

**2.30.1.3** `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 724 of file `stl_algo.h`.

References `std::find_if_not()`.

**2.30.1.4** `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 758 of file `stl_algo.h`.

References `std::none_of()`.

**2.30.1.5** `template<typename _InputIterator , typename _Tp >  
iterator_traits<_InputIterator>::difference_type std::count  
(_InputIterator __first, _InputIterator __last, const _Tp & __value)  
[inline]`

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 4411 of file `stl_algo.h`.

```
2.30.1.6 template<typename _InputIterator , typename _Predicate >
iterator_traits<_InputIterator>::difference_type std::count_if
(_InputIterator __first, _InputIterator __last, _Predicate __pred)
[inline]
```

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 4436 of file `stl_algo.h`.

```
2.30.1.7 template<typename _Iiter1 , typename _Iiter2 , typename
_BinaryPredicate > bool std::equal (_Iiter1 __first1, _Iiter1 __last1,
_Iiter2 __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 991 of file `stl_algobase.h`.

**2.30.1.8** `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 959 of file `stl_algobase.h`.

Referenced by `std::operator==()`.

**2.30.1.9** `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 4223 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

```
2.30.1.10 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 [*first1*,*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 690 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.30.1.11 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 `[first1,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 643 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.30.1.12 template<typename _InputIterator , typename _ForwardIterator
 , typename _BinaryPredicate > _InputIterator std::find_first_of
 (_InputIterator __first1, _InputIterator __last1, _ForwardIterator
 __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
 [inline]
```

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 4316 of file stl\_algo.h.

```
2.30.1.13 template<typename _InputIterator , typename _ForwardIterator
> _InputIterator std::find_first_of (_InputIterator __first1,
_InputIterator __last1, _ForwardIterator __first2, _ForwardIterator
__last2) [inline]
```

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 4276 of file stl\_algo.h.

```
2.30.1.14 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 4247 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

**2.30.1.15** `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 773 of file `stl_algo.h`.

References `std::__find_if_not()`, and `std::__iterator_category()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

**2.30.1.16** `template<typename _InputIterator , typename _Function >  
_Function std::for_each (_InputIterator __first, _InputIterator  
__last, _Function __f) [inline]`

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 4202 of file `stl_algo.h`.

```
2.30.1.17 template<typename _InputIterator1 , typename _InputIterator2
, typename _BinaryPredicate > pair<_InputIterator1,
 _InputIterator2> std::mismatch (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate
 __binary_pred) [inline]
```

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 1134 of file `stl_algobase.h`.

```
2.30.1.18 template<typename _InputIterator1 , typename _InputIterator2
> pair<_InputIterator1, _InputIterator2> std::mismatch
(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2) [inline]
```

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 1096 of file `stl_algobase.h`.

**2.30.1.19** `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 741 of file `stl_algo.h`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

**2.30.1.20** `template<typename _ForwardIterator1 , typename  
_ForwardIterator2 , typename _BinaryPredicate >  
_ForwardIterator1 std::search (_ForwardIterator1 __first1,  
_ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_ForwardIterator2 __last2, _BinaryPredicate __predicate)  
[inline]`

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 4548 of file `stl_algo.h`.

```
2.30.1.21 template<typename _ForwardIterator1 , typename
 _ForwardIterator2 > _ForwardIterator1 std::search
 (_ForwardIterator1 __first1, _ForwardIterator1 __last1,
 _ForwardIterator2 __first2, _ForwardIterator2 __last2) [inline]
```

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 4476 of file `stl_algo.h`.

**2.30.1.22** `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) [inline]`

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 4658 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

**2.30.1.23** `template<typename _ForwardIterator, typename _Integer, typename _Tp > _ForwardIterator std::search_n(_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val) [inline]`

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.



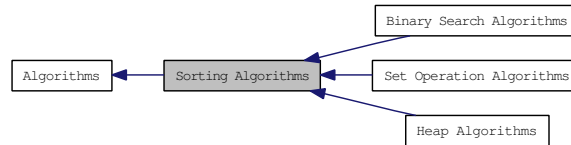
Searches the range `[first,last)` for `count` consecutive elements equal to `val`.

Definition at line 4621 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

## 2.31 Sorting Algorithms

Collaboration diagram for Sorting Algorithms:



### Modules

- [Set Operation Algorithms](#)
- [Binary Search Algorithms](#)
- [Heap Algorithms](#)

### Functions

- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `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 _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 _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `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 _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `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, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `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, 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, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __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 _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)`

### 2.31.1 Function Documentation

**2.31.1.1** `template<typename _BidirectionalIterator, typename _Compare  
> void std::inplace_merge (_BidirectionalIterator __first,  
_BidirectionalIterator __middle, _BidirectionalIterator __last,  
_Compare __comp) [inline]`

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 3180 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()`.

**2.31.1.2** `template<typename _BidirectionalIterator > void std::inplace_merge  
(_BidirectionalIterator __first, _BidirectionalIterator __middle,  
_BidirectionalIterator __last) [inline]`

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 3125 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()`.

**2.31.1.3** `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 3894 of file stl\_algo.h.

References `std::is_sorted_until()`.

**2.31.1.4** `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 3880 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.31.1.5** `template<typename _ForwardIterator, typename _Compare >  
_ForwardIterator std::is_sorted_until (_ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp) [inline]`

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 3937 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

**2.31.1.6** `template<typename _ForwardIterator > _ForwardIterator  
std::is_sorted_until (_ForwardIterator __first, _ForwardIterator  
__last) [inline]`

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 3908 of file `stl_algo.h`.

```
2.31.1.7 template<typename _II1, typename _II2, typename _Compare >
bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2
__first2, _II2 __last2, _Compare __comp) [inline]
```

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 1056 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

```
2.31.1.8 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 1022 of file `stl_algobase.h`.



**2.31.1.9** `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 253 of file `stl_algobase.h`.

**2.31.1.10** `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 209 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()`.

**2.31.1.11** `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp) [inline]`

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 6081 of file stl\_algo.h.

Referenced by std::valarray<\_Tp >::max().

**2.31.1.12** `template<typename _ForwardIterator > _ForwardIterator  
std::max_element (_ForwardIterator __first, _ForwardIterator  
__last) [inline]`

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 6053 of file stl\_algo.h.

**2.31.1.13** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator , typename _Compare > _OutputIterator  
std::merge (_InputIterator1 __first1, _InputIterator1 __last1,  
_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator  
__result, _Compare __comp) [inline]`

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 5359 of file stl\_algo.h.

```
2.31.1.14 template<typename _InputIterator1 , typename _InputIterator2
, typename _OutputIterator > _OutputIterator std::merge
(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)
[inline]
```

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 5296 of file stl\_algo.h.

**2.31.1.15** `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 232 of file `stl_algobase.h`.

**2.31.1.16** `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 186 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__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().

**2.31.1.17** `template<typename _ForwardIterator, typename _Compare >
 _ForwardIterator std::min_element (_ForwardIterator __first,
 _ForwardIterator __last, _Compare __comp) [inline]`

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 6025 of file stl\_algo.h.

Referenced by std::valarray< \_Tp >::min().

**2.31.1.18** `template<typename _ForwardIterator > _ForwardIterator
 std::min_element (_ForwardIterator __first, _ForwardIterator
 __last) [inline]`

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 5997 of file stl\_algo.h.

**2.31.1.19** `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 3985 of file stl\_algo.h.

**2.31.1.20** `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 3966 of file stl\_algo.h.

**2.31.1.21** `template<typename _ForwardIterator , typename _Compare >  
pair<_ForwardIterator, _ForwardIterator> std::minmax_element  
(_ForwardIterator __first, _ForwardIterator __last, _Compare  
__comp) [inline]`

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 4080 of file stl\_algo.h.

**2.31.1.22** `template<typename _ForwardIterator > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator __first, _ForwardIterator __last) [inline]`

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 4004 of file `stl_algo.h`.

**2.31.1.23** `template<typename _BidirectionalIterator, typename _Compare > bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp) [inline]`

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 3639 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.31.1.24** `template<typename _BidirectionalIterator > bool  
std::next_permutation (_BidirectionalIterator __first,  
_BidirectionalIterator __last) [inline]`

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 3582 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.31.1.25** `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 5180 of file `stl_algo.h`.

References `std::__lg()`.

```
2.31.1.26 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. 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 5141 of file `stl_algo.h`.

References `std::__lg()`.

```
2.31.1.27 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  $are$  iterators in the range  $[first, middle)$  such that  $are$  precedes and is an [iterator](#) in the range  $[middle, last)$  then `*comp(j,*i)` and `comp(*k,*i)` are both false.

Definition at line 5103 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

**2.31.1.28** `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  $are$  iterators in the range  $[first, middle)$  such that  $are$  precedes and is an [iterator](#) in the range  $[middle, last)$  then `*j<*i` and `*k<*i` are both false.

Definition at line 5064 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

**2.31.1.29** `template<typename _InputIterator , typename`  
 `_RandomAccessIterator , typename _Compare >`  
 `_RandomAccessIterator std::partial_sort_copy ( _InputIterator`  
 `__first, _InputIterator __last, _RandomAccessIterator __result_first,`  
 `_RandomAccessIterator __result_last, _Compare __comp)`  
`[inline]`

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 2002 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

```
2.31.1.30 template<typename InputIterator , typename
 RandomAccessIterator > RandomAccessIterator
 std::partial_sort_copy (InputIterator first, InputIterator last,
 RandomAccessIterator result_first, RandomAccessIterator
 result_last) [inline]
```

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 *j* are iterators in the range [result\_first,result\_first+N) such that *i* precedes *j* then `*j < *i` is false. The value returned is *result\_first*+N.

Definition at line 1936 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

```
2.31.1.31 template<typename _BidirectionalIterator , typename _Compare
> bool std::prev_permutation (_BidirectionalIterator __first,
 _BidirectionalIterator __last, _Compare __comp) [inline]
```

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 3752 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

```
2.31.1.32 template<typename _BidirectionalIterator > bool
std::prev_permutation (_BidirectionalIterator __first,
 _BidirectionalIterator __last) [inline]
```

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 3695 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

```
2.31.1.33 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 5254 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

```
2.31.1.34 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 5218 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

**2.31.1.35** `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 5460 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

**2.31.1.36** `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 5418 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

## 2.32 Set Operation Algorithms

Collaboration diagram for Set Operation Algorithms:



### Functions

- `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 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, InputIterator2 __last2)`
- `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, 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, type-`  
`name _Compare >`  
`_OutputIterator std::set\_union (_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)`

### 2.32.1 Detailed Description

These algorithms are common [set](#) operations performed on sequences that are already sorted. The number of comparisons will be linear.

### 2.32.2 Function Documentation

**2.32.2.1** `template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _Compare > bool std::includes (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
 __last2, _Compare __comp) [inline]`

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 3528 of file `stl_algo.h`.

```
2.32.2.2 template<typename _InputIterator1 , typename _InputIterator2 >
bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1,
 _InputIterator2 __first2, _InputIterator2 __last2) [inline]
```

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 3478 of file stl\_algo.h.

```
2.32.2.3 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) [inline]
```

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 5817 of file `stl_algo.h`.

```
2.32.2.4 template<typename _InputIterator1 , typename _InputIterator2 ,
typename _OutputIterator > _OutputIterator std::set_difference
(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)
[inline]
```

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 5756 of file `stl_algo.h`.

```
2.32.2.5 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) [inline]
```

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 5698 of file `stl_algo.h`.

```
2.32.2.6 template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _OutputIterator > _OutputIterator std::set_intersection
 (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
 __first2, _InputIterator2 __last2, _OutputIterator __result)
 [inline]
```

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 5641 of file `stl_algo.h`.

```
2.32.2.7 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) [inline]
```

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 5941 of file `stl_algo.h`.

```
2.32.2.8 template<typename _InputIterator1 , typename
 _InputIterator2 , typename _OutputIterator > _OutputIterator
std::set_symmetric_difference (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
 __last2, _OutputIterator __result) [inline]
```

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 5875 of file `stl_algo.h`.

```
2.32.2.9 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) [inline]
```

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 5574 of file `stl_algo.h`.

```
2.32.2.10 template<typename _InputIterator1, typename _InputIterator2
, typename _OutputIterator > _OutputIterator std::set_union
(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)
[inline]
```

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 5507 of file `stl_algo.h`.

## 2.33 Binary Search Algorithms

Collaboration diagram for Binary Search Algorithms:



### Functions

- `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 >`  
`bool std::binary\_search (_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 _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)`

### 2.33.1 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.

### 2.33.2 Function Documentation

**2.33.2.1** `template<typename _ForwardIterator, typename _Tp, typename _Compare > bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp) [inline]`

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 2765 of file `stl_algo.h`.

References `std::lower_bound()`.

**2.33.2.2** `template<typename _ForwardIterator, typename _Tp > bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val) [inline]`

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 2732 of file `stl_algo.h`.

References `std::lower_bound()`.

**2.33.2.3** `template<typename _ForwardIterator , typename _Tp , typename  
 _Compare > pair<_ForwardIterator, _ForwardIterator>  
 std::equal_range (_ForwardIterator __first, _ForwardIterator __last,  
 const _Tp & __val, _Compare __comp) [inline]`

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 2671 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

```
2.33.2.4 template<typename _ForwardIterator, typename _Tp >
pair<_ForwardIterator, _ForwardIterator> std::equal_range
(_ForwardIterator __first, _ForwardIterator __last, const _Tp &
__val) [inline]
```

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 2609 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

```
2.33.2.5 template<typename _ForwardIterator, typename _Tp, typename
_Compare > _ForwardIterator std::lower_bound (_ForwardIterator
__first, _ForwardIterator __last, const _Tp & __val, _Compare
__comp) [inline]
```

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 2454 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

**2.33.2.6** `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator std::lower_bound (_ForwardIterator __first,  
_ForwardIterator __last, const _Tp & __val) [inline]`

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 2402 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

**2.33.2.7** `template<typename _ForwardIterator, typename _Tp, typename  
_Compare > _ForwardIterator std::upper_bound (_ForwardIterator  
__first, _ForwardIterator __last, const _Tp & __val, _Compare  
__comp) [inline]`

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 2554 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, and `std::equal_range()`.

```
2.33.2.8 template<typename _ForwardIterator , typename _Tp >
 _ForwardIterator std::upper_bound (_ForwardIterator __first,
 _ForwardIterator __last, const _Tp & __val) [inline]
```

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 2503 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

## 2.34 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>.*

### 2.34.1 Detailed Description

Classes encapsulating memory operations.

## 2.35 Atomics

### 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< void * >`  
*Explicit specialization for void\*.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Defines

- #define `_ATOMIC_CMPEXCHNG`(\_\_a, \_\_e, \_\_m, \_\_x)
- #define `_ATOMIC_LOAD`(\_\_a, \_\_x)
- #define `_ATOMIC_MODIFY`(\_\_a, \_\_o, \_\_m, \_\_x)
- #define `_ATOMIC_STORE`(\_\_a, \_\_m, \_\_x)
- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

## Typedefs

- typedef struct `std::__atomic_flag_base` **`std::__atomic_flag_base`**
- 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_short` **`std::atomic_int_fast16_t`**
- typedef `atomic_int` **`std::atomic_int_fast32_t`**
- typedef `atomic_llong` **`std::atomic_int_fast64_t`**
- typedef `atomic_schar` **`std::atomic_int_fast8_t`**
- typedef `atomic_short` **`std::atomic_int_least16_t`**
- typedef `atomic_int` **`std::atomic_int_least32_t`**

- typedef [atomic\\_llong](#) `std::atomic_int_least64_t`
- typedef [atomic\\_schar](#) `std::atomic_int_least8_t`
- typedef [atomic\\_llong](#) `std::atomic_intmax_t`
- typedef [atomic\\_long](#) `std::atomic_intptr_t`
- typedef `__atomic_base< long long >` [atomic\\_llong](#)
- typedef `__atomic_base< long >` [atomic\\_long](#)
- typedef [atomic\\_long](#) `std::atomic_ptrdiff_t`
- typedef `__atomic_base< signed char >` [atomic\\_schar](#)
- typedef `__atomic_base< short >` [atomic\\_short](#)
- typedef [atomic\\_ulong](#) `std::atomic_size_t`
- typedef [atomic\\_long](#) `std::atomic_ssize_t`
- typedef `__atomic_base< unsigned char >` [atomic\\_uchar](#)
- typedef `__atomic_base< unsigned int >` [atomic\\_uint](#)
- typedef [atomic\\_ushort](#) `std::atomic_uint_fast16_t`
- typedef [atomic\\_uint](#) `std::atomic_uint_fast32_t`
- typedef [atomic\\_ullong](#) `std::atomic_uint_fast64_t`
- typedef [atomic\\_uchar](#) `std::atomic_uint_fast8_t`
- typedef [atomic\\_ushort](#) `std::atomic_uint_least16_t`
- typedef [atomic\\_uint](#) `std::atomic_uint_least32_t`
- typedef [atomic\\_ullong](#) `std::atomic_uint_least64_t`
- typedef [atomic\\_uchar](#) `std::atomic_uint_least8_t`
- typedef [atomic\\_ullong](#) `std::atomic_uintmax_t`
- typedef [atomic\\_ulong](#) `std::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 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

- void **std::atomic\_flag\_wait\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) throw ()
- **std::atomic**\_\_ ((\_\_const\_\_)) \_\_atomic\_flag\_base \*\_\_atomic\_flag\_for\_address(const void \*\_\_z) throw ()
- memory\_order **std::calculate\_memory\_order** (memory\_order \_\_m)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_strong** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool **std::atomic\_compare\_exchange\_strong** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- bool **std::atomic\_compare\_exchange\_strong** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, 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** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_weak** (\_\_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)
- 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** (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)
- template<typename \_ITp >  
\_ITp **std::atomic\_exchange** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- bool **std::atomic\_exchange** (atomic\_bool \*\_\_a, bool \_\_i)
- void \* **std::atomic\_exchange** (atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
\_ITp **std::atomic\_exchange\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- bool **std::atomic\_exchange\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)

- void \* **std::atomic\_exchange\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- template<typename \_ITp >  
\_ITp **std::atomic\_fetch\_add** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void \* **std::atomic\_fetch\_add** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- template<typename \_ITp >  
\_ITp **std::atomic\_fetch\_add\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void \* **std::atomic\_fetch\_add\_explicit** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, 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\_explicit** (\_\_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\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- template<typename \_ITp >  
\_ITp **std::atomic\_fetch\_sub** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void \* **std::atomic\_fetch\_sub** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- template<typename \_ITp >  
\_ITp **std::atomic\_fetch\_sub\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, 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\_xor** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- 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\_base \*\_\_a)
- void **std::atomic\_flag\_clear\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) throw ()
- void **std::atomic\_flag\_clear\_explicit** (atomic\_flag \*\_\_a, memory\_order \_\_m)
- bool **std::atomic\_flag\_test\_and\_set** (\_\_atomic\_flag\_base \*\_\_a)
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) throw ()
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (atomic\_flag \*\_\_a, memory\_order \_\_m)
- template<typename \_ITp >  
bool **std::atomic\_is\_lock\_free** (const \_\_atomic\_base< \_ITp > \*\_\_a)

- bool **std::atomic\_is\_lock\_free** (const atomic\_bool \*\_\_a)
- bool **std::atomic\_is\_lock\_free** (const atomic\_address \*\_\_a)
- template<typename \_ITp >  
\_ITp **std::atomic\_load** (const \_\_atomic\_base< \_ITp > \*\_\_a)
- bool **std::atomic\_load** (const atomic\_bool \*\_\_a)
- void \* **std::atomic\_load** (const atomic\_address \*\_\_a)
- template<typename \_ITp >  
\_ITp **std::atomic\_load\_explicit** (const \_\_atomic\_base< \_ITp > \*\_\_a, memory\_order \_\_m)
- 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 >  
void **std::atomic\_store** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void **std::atomic\_store** (atomic\_bool \*\_\_a, bool \_\_i)
- void **std::atomic\_store** (atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, 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\_seq\_cst)
- bool **std::atomic< \_Tp \* >::compare\_exchange\_strong** (\_Tp \*&, \_Tp \*, memory\_order, memory\_order)
- 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)
- \_Tp \* **std::atomic< \_Tp \* >::exchange** (\_Tp \*, memory\_order=memory\_order\_seq\_cst)
- \_Tp \* **std::atomic< \_Tp \* >::fetch\_add** (ptrdiff\_t, memory\_order=memory\_order\_seq\_cst)
- \_Tp \* **std::atomic< \_Tp \* >::fetch\_sub** (ptrdiff\_t, memory\_order=memory\_order\_seq\_cst)
- template<typename \_Tp >  
\_Tp **std::kill\_dependency** (\_Tp \_\_y)
- \_Tp \* **std::atomic< \_Tp \* >::load** (memory\_order=memory\_order\_seq\_cst) const

### 2.35.1 Detailed Description

Components for performing [atomic](#) operations.

### 2.35.2 Define Documentation

#### 2.35.2.1 `#define _GLIBCXX_ATOMIC_PROPERTY`

29.2 Lock-free Property

Definition at line 68 of file `atomic_base.h`.

### 2.35.3 Typedef Documentation

#### 2.35.3.1 `typedef __atomic_base<char> atomic_char`

`atomic_char`

Definition at line 71 of file `atomicfwd_cxx.h`.

#### 2.35.3.2 `typedef __atomic_base<char16_t> atomic_char16_t`

`atomic_char16_t`

Definition at line 107 of file `atomicfwd_cxx.h`.

#### 2.35.3.3 `typedef __atomic_base<char32_t> atomic_char32_t`

`atomic_char32_t`

Definition at line 110 of file `atomicfwd_cxx.h`.

#### 2.35.3.4 `typedef __atomic_base<int> atomic_int`

`atomic_int`

Definition at line 86 of file `atomicfwd_cxx.h`.

#### 2.35.3.5 `typedef __atomic_base<long long> atomic_llong`

`atomic_llong`

Definition at line 98 of file `atomicfwd_cxx.h`.

**2.35.3.6 typedef \_\_atomic\_base<long> atomic\_long**

atomic\_long

Definition at line 92 of file atomicfwd\_cxx.h.

**2.35.3.7 typedef \_\_atomic\_base<signed char> atomic\_schar**

atomic\_schar

Definition at line 74 of file atomicfwd\_cxx.h.

**2.35.3.8 typedef \_\_atomic\_base<short> atomic\_short**

atomic\_short

Definition at line 80 of file atomicfwd\_cxx.h.

**2.35.3.9 typedef \_\_atomic\_base<unsigned char> atomic\_uchar**

atomic\_uchar

Definition at line 77 of file atomicfwd\_cxx.h.

**2.35.3.10 typedef \_\_atomic\_base<unsigned int> atomic\_uint**

atomic\_uint

Definition at line 89 of file atomicfwd\_cxx.h.

**2.35.3.11 typedef \_\_atomic\_base<unsigned long long> atomic\_ullong**

atomic\_ullong

Definition at line 101 of file atomicfwd\_cxx.h.

**2.35.3.12 typedef \_\_atomic\_base<unsigned long> atomic\_ulong**

atomic\_ulong

Definition at line 95 of file atomicfwd\_cxx.h.

**2.35.3.13 typedef \_\_atomic\_base<unsigned short> atomic\_ushort**

atomic\_ushort

Definition at line 83 of file atomicfwd\_cxx.h.

**2.35.3.14 typedef \_\_atomic\_base<wchar\_t> atomic\_wchar\_t**

atomic\_wchar\_t

Definition at line 104 of file atomicfwd\_cxx.h.

**2.35.3.15 typedef enum std::memory\_order std::memory\_order**

Enumeration for memory\_order.

**2.35.4 Enumeration Type Documentation****2.35.4.1 enum std::memory\_order**

Enumeration for memory\_order.

Definition at line 47 of file atomic\_base.h.

**2.35.5 Function Documentation****2.35.5.1 template<typename \_Tp > \_Tp std::kill\_dependency (\_Tp \_\_y)  
[inline]**

kill\_dependency

Definition at line 54 of file atomic.



## 2.36 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)`

#### 2.36.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

## 2.37 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 `cctype` 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 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.*
- 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::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`.*

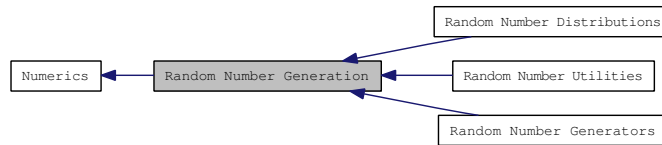
*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

### 2.37.1 Detailed Description

Classes and functions for internationalization and localization.

## 2.38 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)`

#### 2.38.1 Detailed Description

A facility for generating random numbers on selected distributions.

#### 2.38.2 Function Documentation

**2.38.2.1** `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate\_canonical (_UniformRandomNumberGenerator & __g)`  
**[inline]**

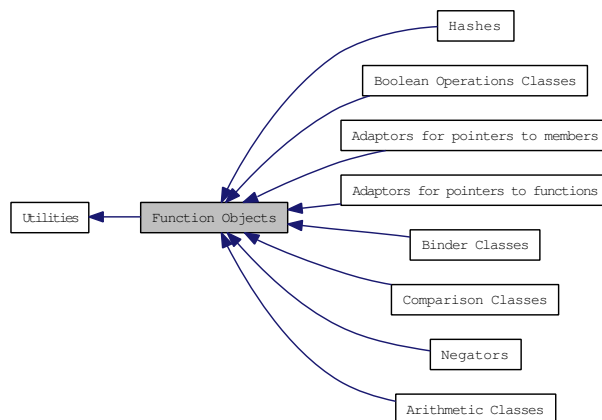
A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 2735 of file `random.tcc`.

References `std::log()`, and `std::min()`.

## 2.39 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)`

### 2.39.1 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.

### 2.39.2 Function Documentation

#### 2.39.2.1 `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 766 of file `functional`.

## 2.40 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.*

### 2.40.1 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.



## 2.41 Comparison Classes

Collaboration diagram for Comparison Classes:



### Classes

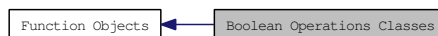
- struct `std::equal_to< _Tp >`  
*One of the `comparison` functors.*
- struct `std::greater< _Tp >`  
*One of the `comparison` functors.*
- struct `std::greater_equal< _Tp >`  
*One of the `comparison` functors.*
- struct `std::less< _Tp >`  
*One of the `comparison` functors.*
- struct `std::less_equal< _Tp >`  
*One of the `comparison` functors.*
- struct `std::not_equal_to< _Tp >`  
*One of the `comparison` functors.*

### 2.41.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

## 2.42 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.*

### 2.42.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

## 2.43 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)`

#### 2.43.1 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.

## 2.43.2 Function Documentation

**2.43.2.1** `template<typename _Predicate > unary_negate<_Predicate>  
std::not1 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 364 of file `stl_function.h`.

**2.43.2.2** `template<typename _Predicate > binary_negate<_Predicate>  
std::not2 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 389 of file `stl_function.h`.

## 2.44 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 _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_-`  
`Result(*__x)(_Arg1, _Arg2))`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(-`  
`Arg))`

#### 2.44.1 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.

## 2.44.2 Function Documentation

**2.44.2.1** `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 463 of file `stl_function.h`.

**2.44.2.2** `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 437 of file `stl_function.h`.

## 2.45 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, typename _Arg >`  
`mem_fun1_t<_Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*_f)(_Arg))`
- `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_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)())`

### 2.45.1 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.



## 2.46 Heap Algorithms

Collaboration diagram for Heap Algorithms:



### Functions

- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `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)`

## 2.46.1 Function Documentation

**2.46.1.1** `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 571 of file `stl_heap.h`.

References `std::is_heap_until()`.

**2.46.1.2** `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 558 of file `stl_heap.h`.

References `std::is_heap_until()`.

```
2.46.1.3 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 536 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

```
2.46.1.4 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 510 of file `stl_heap.h`.

References `std::distance()`.

```
2.46.1.5 template<typename _RandomAccessIterator , typename _Compare
> void std::make_heap (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp) [inline]
```

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 413 of file stl\_heap.h.

Referenced by std::\_\_heap\_select(), std::partial\_sort\_copy(), and std::priority\_queue<\_Tp, \_Sequence, \_Compare >::priority\_queue().

```
2.46.1.6 template<typename _RandomAccessIterator > void std::make_heap
(_RandomAccessIterator __first, _RandomAccessIterator __last)
[inline]
```

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 373 of file stl\_heap.h.

```
2.46.1.7 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 350 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`.

```
2.46.1.8 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 276 of file `stl_heap.h`.

```
2.46.1.9 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 203 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

```
2.46.1.10 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 154 of file stl\_heap.h.

**2.46.1.11** `template<typename _RandomAccessIterator , typename  
_Compare > void std::sort_heap (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp) [inline]`

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 481 of file stl\_heap.h.

Referenced by std::partial\_sort(), and std::partial\_sort\_copy().

**2.46.1.12** `template<typename _RandomAccessIterator > void std::sort_heap  
(_RandomAccessIterator __first, _RandomAccessIterator __last)  
[inline]`

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 452 of file stl\_heap.h.

## 2.47 Iterators

### Classes

- class `std::back_insert_iterator< _Container >`  
*Turns assignment into insertion.*
- 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.*
- 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< _Iterator >`  
*Traits class for iterators.*
- 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.
- struct `std::output_iterator_tag`  
Marking output iterators.
- struct `std::random_access_iterator_tag`  
Random-access iterators support a superset of bidirectional *iterator* operations.
- class `std::reverse_iterator<_Iterator >`

## Functions

- 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<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, ostreambuf_iterator< _CharT > >::__type`  
`std::__copy_move_a2` (\_CharT \* \_\_first, \_CharT \* \_\_last, ostreambuf\_iterator< \_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 _CharT , typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `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 _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 >`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_`  
`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 _IteratorL , typename _IteratorR >`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `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 _Iterator >`  
`reverse_iterator< _Iterator >::difference_type 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 _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 _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 _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `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 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 >`  
`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 _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 _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)`

### 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.

### 2.47.2 Function Documentation

#### 2.47.2.1 `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 168 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()`.

#### 2.47.2.2 `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 464 of file `stl_iterator.h`.

Referenced by `std::regex_replace()`.

**2.47.2.3** `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 547 of file `stl_iterator.h`.

**2.47.2.4** `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 653 of file `stl_iterator.h`.

**2.47.2.5** `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 135 of file `stream_iterator.h`.

**2.47.2.6** `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 128 of file stream\_iterator.h.

**2.47.2.7** `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.

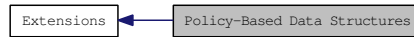
Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 283 of file stl\_iterator.h.

References `std::reverse_iterator< _Iterator >::base()`.

## 2.48 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`

### 2.48.1 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](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

## 2.49 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 _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)`

### 2.49.1 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 2.1: Random Number Generator Requirements

### 2.49.2 Typedef Documentation

#### 2.49.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1341 of file `random.h`.

#### 2.49.2.2 `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 1335 of file `random.h`.

**2.49.2.3** `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 1357 of file random.h.

**2.49.2.4** `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1369 of file random.h.

### 2.49.3 Function Documentation

**2.49.3.1** `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) [inline]`

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.

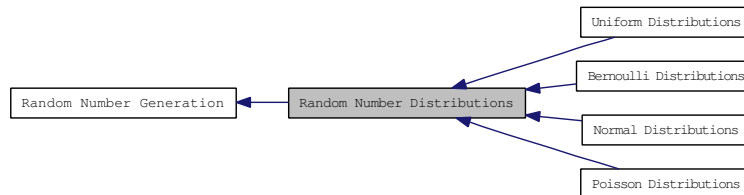
#### Returns:

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1109 of file random.h.

## 2.50 Random Number Distributions

Collaboration diagram for Random Number Distributions:



### Modules

- [Uniform Distributions](#)
- [Normal Distributions](#)
- [Bernoulli Distributions](#)
- [Poisson Distributions](#)

## 2.51 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 _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 _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> &)`

### 2.51.1 Function Documentation

- 2.51.1.1** `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) [inline]`

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 890 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

```
2.51.1.2 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) [inline]
```

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 848 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

```
2.51.1.3 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) [inline]
```

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 914 of file `random.tcc`.

References `std::ios_base::flags()`, `std::uniform_real_distribution< _RealType >::param()`, and `std::skipws()`.

**2.51.1.4** `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) [inline]`

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 869 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution< _IntType >::param()`, and `std::skipws()`.

## 2.52 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, typename _CharT, typename _Traits>  
std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::cauchy_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> &)`

## 2.52.1 Function Documentation

**2.52.1.1** `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) [inline]`

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 1792 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.52.1.2** `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) [inline]`

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 1816 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.



## 2.53 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

- `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 &)`
- `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)`

### 2.53.1 Function Documentation

**2.53.1.1** `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) [inline]`

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 985 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.53.1.2** `template<typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<  
(std::basic_ostream< _CharT, _Traits > & __os, const  
std::bernoulli_distribution & __x) [inline]`

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 935 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.53.1.3** `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) [inline]`

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 1009 of file `random.tcc`.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType >::param()`, and `std::skipws()`.

**2.53.1.4** `template<typename _CharT , typename _Traits >  
std::basic_istream< _CharT, _Traits>& std::operator>>  
(std::basic_istream< _CharT, _Traits > & __is,  
std::bernoulli_distribution & __x) [inline]`

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 3005 of file `random.h`.

References `std::bernoulli_distribution::param()`.

## 2.54 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 _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, 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_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_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::exponential_distribution< _RealType > &)`

### 2.54.1 Function Documentation

**2.54.1.1** `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) [inline]`

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 2102 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.54.1.2** `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) [inline]`

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 2045 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.54.1.3** `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) [inline]`

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 1545 of file random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.54.1.4** `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) [inline]`

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 2126 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution< _RealType >::param()`, and `std::skipws()`.

**2.54.1.5** `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) [inline]`

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 2069 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution< _RealType >::param()`, and `std::skipws()`.

**2.54.1.6** `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) [inline]`

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 1568 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution<_RealType>::param()`, and `std::skipws()`.



## 2.55 Random Number Utilities

Collaboration diagram for Random Number Utilities:



### Classes

- class [std::seed\\_seq](#)

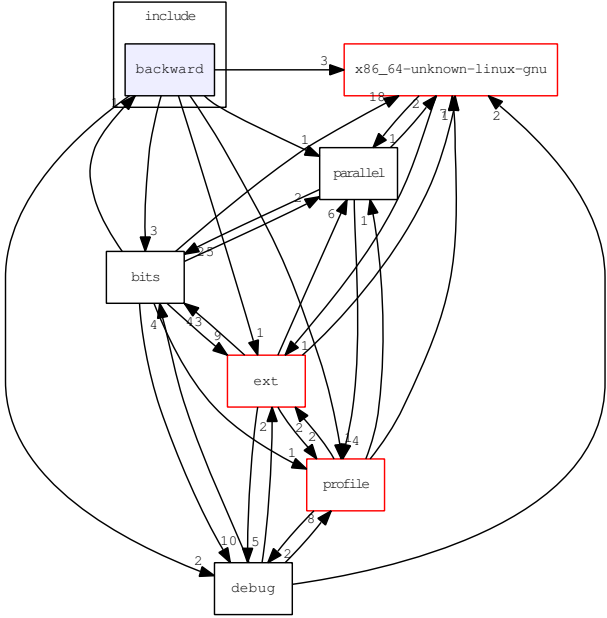
*The [seed\\_seq](#) class generates sequences of seeds for random number generators.*



# Chapter 3

# Directory Documentation

## 3.1 include/backward/ Directory Reference

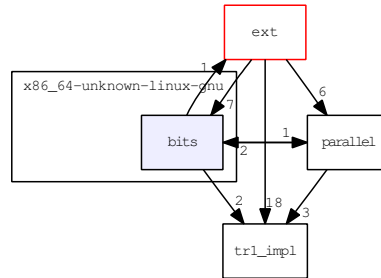


### Files

- 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 **sstream**

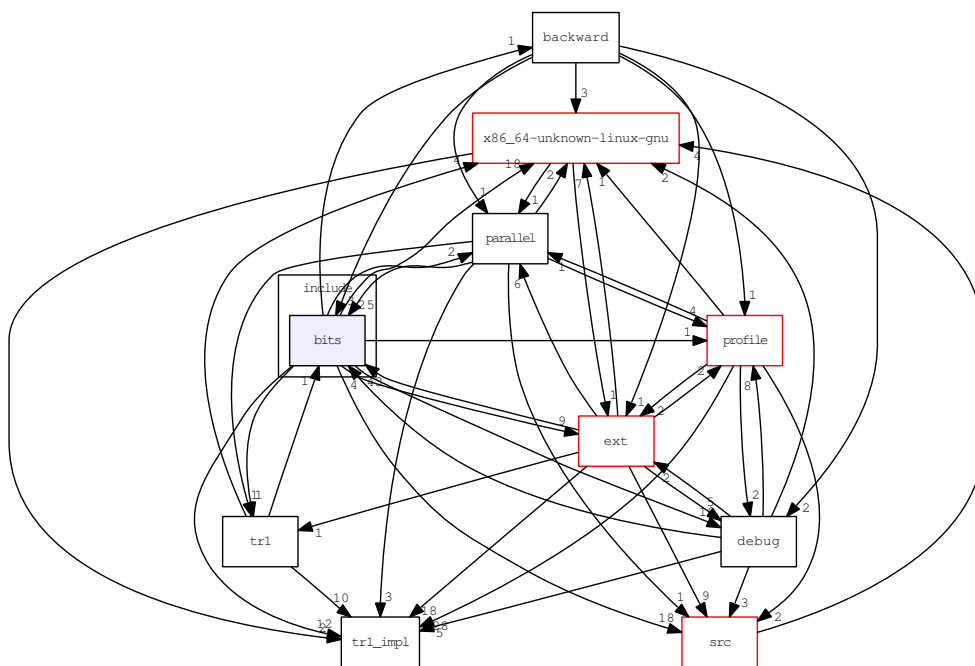
## 3.2 include/x86\_64-unknown-linux-gnu/bits/ Directory Reference



### 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 **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 [time\\_members.h](#)

### 3.3 include/bits/ Directory Reference



#### Files

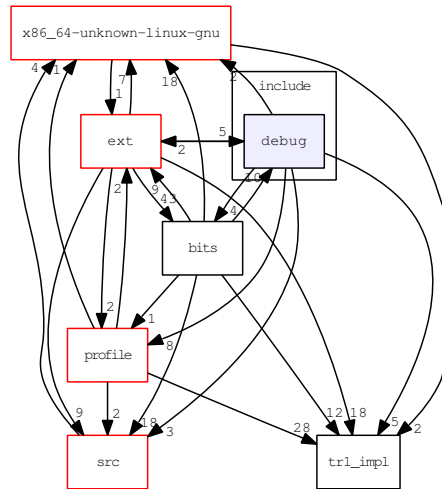
- file [bits/algorithmfwd.h](#)
- file [allocator.h](#)
- file [atomic\\_0.h](#)
- file [atomic\\_2.h](#)
- file [atomic\\_base.h](#)
- file [atomicfwd\\_c.h](#)
- file [atomicfwd\\_cxx.h](#)
- file [basic\\_ios.h](#)
- file [basic\\_ios.tcc](#)
- file [basic\\_string.h](#)
- file [basic\\_string.tcc](#)
- file [boost\\_concept\\_check.h](#)
- file [c++0x\\_warning.h](#)
- file [char\\_traits.h](#)
- file [cmath.tcc](#)
- file [codecvt.h](#)

- file [concept\\_check.h](#)
- file [cpp\\_type\\_traits.h](#)
- file [deque.tcc](#)
- 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 [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 [ostream.tcc](#)
- file [ostream\\_insert.h](#)
- file [postypes.h](#)
- file [random.h](#)
- file [random.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](#)



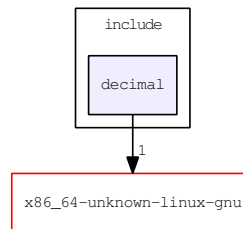
### 3.4 include/debug/ Directory Reference



#### Files

- file [debug/bitset](#)
- file [debug.h](#)
- file [debug/deque](#)
- file [formatter.h](#)
- 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 [debug/set](#)
- file [debug/set.h](#)
- file [debug/string](#)
- file [debug/unordered\\_map](#)
- file [debug/unordered\\_set](#)
- file [debug/vector](#)

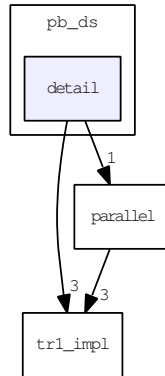
### 3.5 include/decimal/ Directory Reference



#### Files

- file [decimal](#)

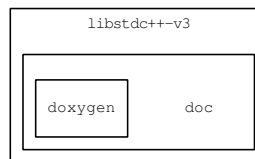
### 3.6 include/ext/pb\_ds/detail/ Directory Reference



#### 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](#)

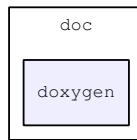
### 3.7 /mnt/share/src/gcc-trunk/libstdc++-v3/doc/ Directory Reference



#### Directories

- directory [doxygen](#)

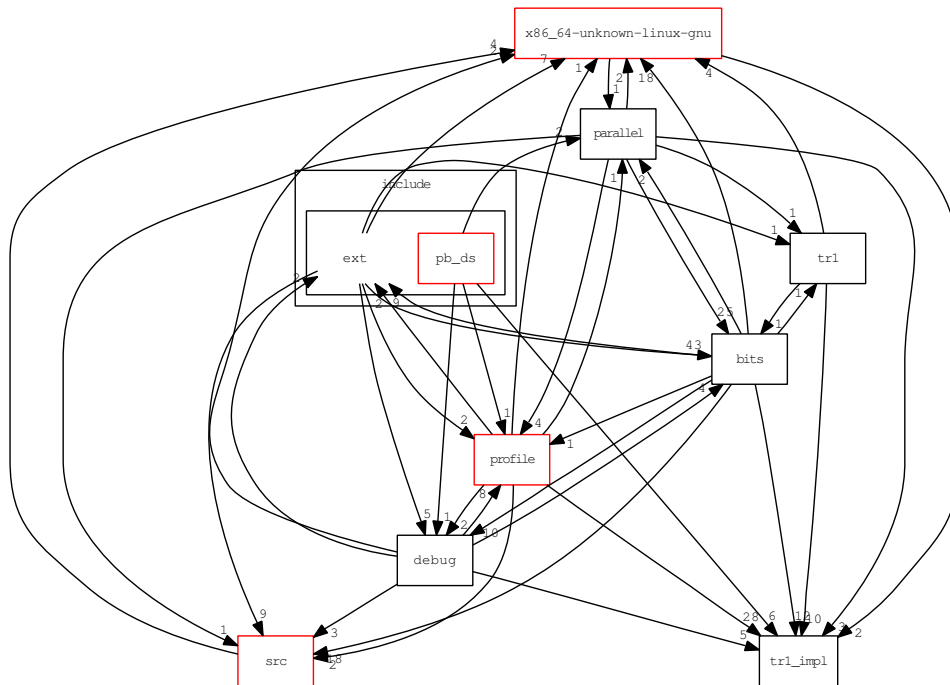
### 3.8 /mnt/share/src/gcc-trunk/libstdc++-v3/doc/doxygen/ Directory Reference



#### Files

- file `doxygroups.cc`

### 3.9 include/ext/ Directory Reference



#### Directories

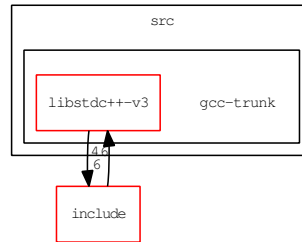
- directory [pb\\_ds](#)

#### Files

- 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-trunk/ Directory Reference

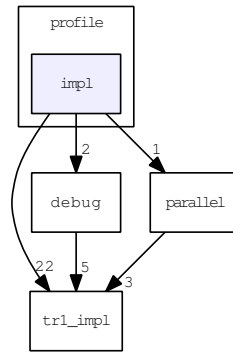


#### Directories

- directory [libstdc++-v3](#)



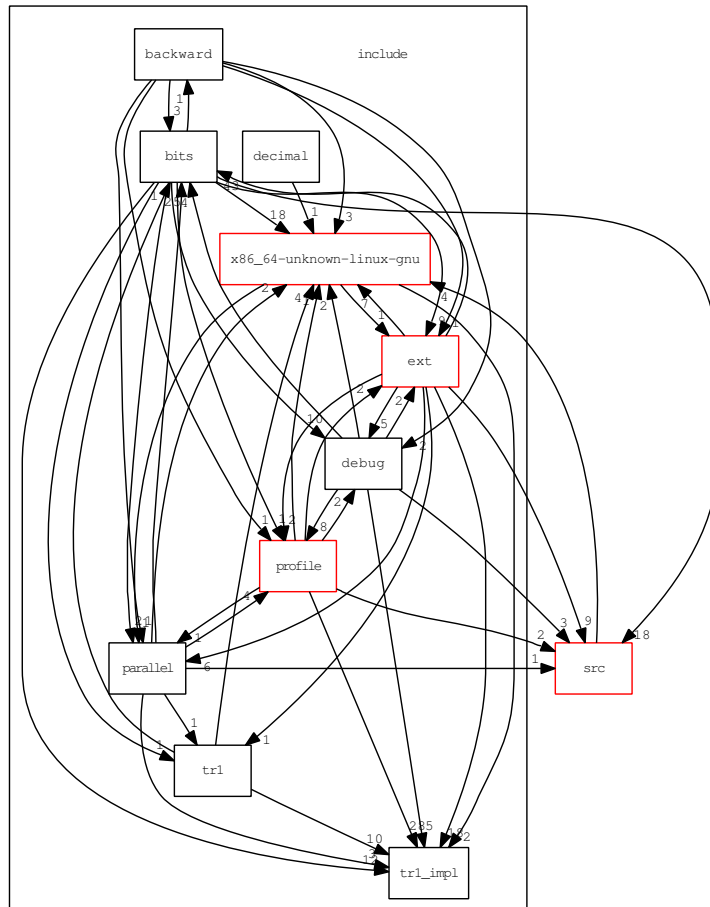
## 3.11 include/profile/impl/ Directory Reference



### Files

- file [profiler.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



#### Directories

- directory [backward](#)
- directory [bits](#)
- directory [debug](#)
- directory [decimal](#)
- directory [ext](#)
- directory [parallel](#)
- directory [profile](#)
- directory [tr1](#)

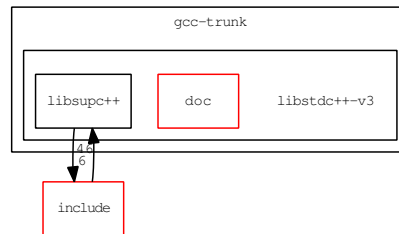
- directory [tr1\\_impl](#)
- directory [x86\\_64-unknown-linux-gnu](#)

## 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 [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 [stdatomic.h](#)
- file [stdexcept](#)
- file [streambuf](#)
- file [string](#)
- file [system\\_error](#)
- file [tgmath.h](#)
- file [thread](#)
- file [tuple](#)
- file [type\\_traits](#)
- file [unordered\\_map](#)
- file [unordered\\_set](#)
- file [utility](#)
- file [valarray](#)
- file [vector](#)

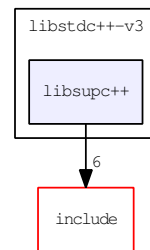
### 3.13 /mnt/share/src/gcc-trunk/libstdc++-v3/ Directory Reference



#### Directories

- directory [doc](#)
- directory [libsupc++](#)

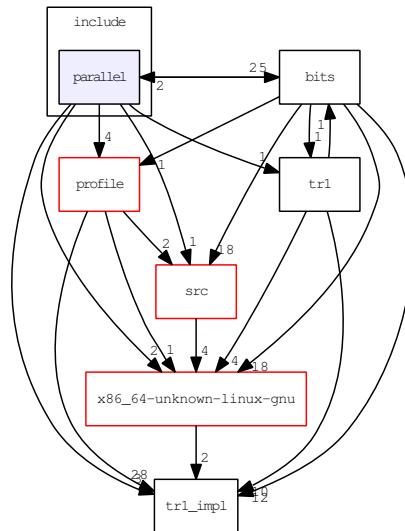
### 3.14 /mnt/share/src/gcc-trunk/libstdc++-v3/libsupc++/ Directory Reference



#### Files

- file [cxxabi-forced.h](#)
- file [cxxabi.h](#)
- file [exception](#)
- file [exception\\_ptr.h](#)
- file [initializer\\_list](#)
- file [nested\\_exception.h](#)
- file [new](#)
- file [typeinfo](#)

### 3.15 include/parallel/ Directory Reference



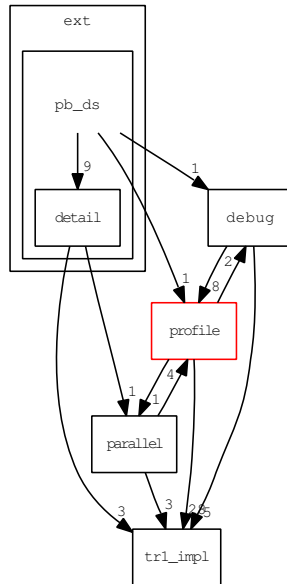
#### Files

- file [algo.h](#)
- file [algorithmbase.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



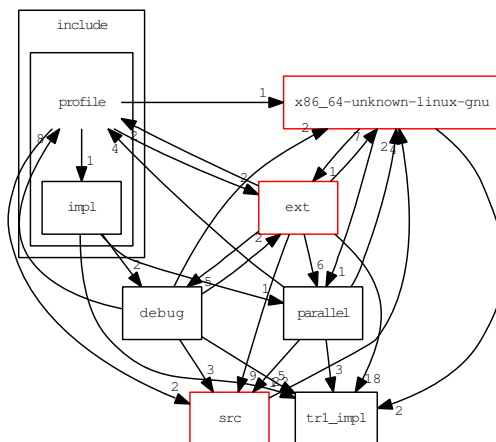
### Directories

- 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



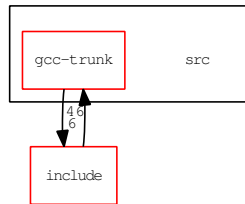
#### Directories

- directory [impl](#)

#### Files

- file [profile/base.h](#)
- file [profile/bitset](#)
- file [profile/deque](#)
- 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](#)

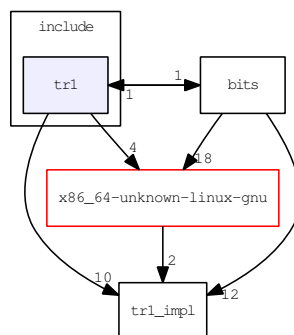
## 3.18 /mnt/share/src/ Directory Reference



### Directories

- directory [gcc-trunk](#)

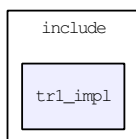
### 3.19 include/tr1/ Directory Reference



#### 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/stdio](#)
- file [tr1/stdlib](#)
- file [tr1/ctgmath](#)
- file [tr1/ctime](#)
- file [tr1/cwchar](#)
- file [tr1/cwctype](#)

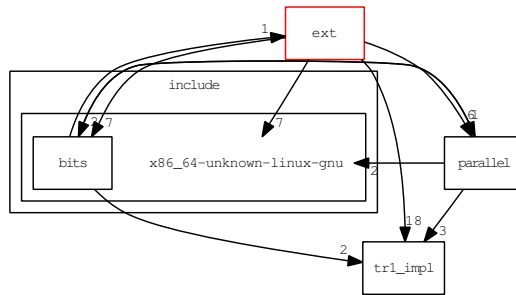
## 3.20 include/tr1\_impl/ Directory Reference



### Files

- file [tr1\\_impl/array](#)
- file [boost\\_sp\\_counted\\_base.h](#)
- file [tr1\\_impl/cctype](#)
- file [tr1\\_impl/cfenv](#)
- file [tr1\\_impl/cinttypes](#)
- file [tr1\\_impl/cmath](#)
- file [tr1\\_impl/complex](#)
- file [tr1\\_impl/cstdint](#)
- file [tr1\\_impl/cstdio](#)
- file [tr1\\_impl/cstdlib](#)
- file [tr1\\_impl/cwchar](#)
- file [tr1\\_impl/cwctype](#)
- file [tr1\\_impl/regex](#)
- file [tr1\\_impl/type\\_traits](#)
- file [tr1\\_impl/utility](#)

### 3.21 include/x86\_64-unknown-linux-gnu/ Directory Reference



#### Directories

- directory [bits](#)

# Chapter 4

## Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

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 [icov](#) 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[stmt.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` (`_FromType` \*`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__const_pointer_cast` (const `_FromType` &`__arg`)
- 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 `_Size`, typename `_OutputIterator` >  
`pair`< `_InputIterator`, `_OutputIterator` > `__copy_n` (`_InputIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `input_iterator_tag`)
- template<typename `_RandomAccessIterator`, typename `_Distance` >  
void `__distance` (`_RandomAccessIterator` `__first`, `_RandomAccessIterator` `__last`, `_Distance` &`__n`, `std::random_access_iterator_tag`)
- template<typename `_InputIterator`, typename `_Distance` >  
void `__distance` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_Distance` &`__n`, `std::input_iterator_tag`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (`_FromType` \*`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (const `_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`)
- else return `__exchange_and_add_single` (`__mem`, `__val`)
- static `_Atomic_word` `__exchange_and_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- template<class `_Concept` >  
void `__function_requires` ()
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type`)
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type` \*`__ptr`)
- int `__lexicographical_compare_3way` (const char \*`__first1`, const char \*`__last1`, const char \*`__first2`, const char \*`__last2`)

- `int __lexicographical_compare_3way` (const unsigned char \*\_\_first1, const unsigned char \*\_\_last1, const unsigned char \*\_\_first2, const unsigned char \*\_\_last2)
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __lexicographical_compare_3way` (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_InputIterator2 \_\_last2)
- `template<typename _Tp, typename _Compare >`  
`const _Tp & __median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c, \_Compare \_\_comp)
- `template<typename _Tp >`  
`const _Tp & __median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c)
- `crope::reference __mutable_reference_at` ([crope](#) &\_\_c, size\_t \_\_i)
- `template<typename _Tp, typename _Integer >`  
`_Tp __power` (\_Tp \_\_x, \_Integer \_\_n)
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp __power` (\_Tp \_\_x, \_Integer \_\_n, \_MonoidOperation \_\_monoid\_op)
- `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` (\_FromType \*\_\_arg)
- `template<typename _ToType, typename _FromType >`  
`_ToType __reinterpret_pointer_cast` (const \_FromType &\_\_arg)
- `_Slist_node_base * __slist_make_link` (\_Slist\_node\_base \*\_\_prev\_node, \_Slist\_node\_base \*\_\_new\_node)
- `const _Slist_node_base * __slist_previous` (const \_Slist\_node\_base \*\_\_head, const \_Slist\_node\_base \*\_\_node)
- `_Slist_node_base * __slist_previous` (\_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)
- `template<typename _ToType, typename _FromType >`  
`_ToType __static_pointer_cast` (\_FromType \*\_\_arg)
- `template<typename _ToType, typename _FromType >`  
`_ToType __static_pointer_cast` (const \_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_broadcast_error` ()
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_concurrency_wait_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)
- `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, [std::input\\_iterator\\_tag](#))
- `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 >](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (\_InputIter \_\_first,  
\_Size \_\_count, \_ForwardIter \_\_result, \_Allocator \_\_alloc)
- `void __verbose_terminate_handler` ()
- `size_t __Bit_scan_forward` (size\_t \_\_num)
- `template<typename _ForwardIterator, typename _Tp >`  
`void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [alloca-  
tor< \\_Tp >](#))
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_-  
Allocator \_\_alloc)
- `template<class _CharT, class _Traits >`  
`void __Rope_fill` ([basic\\_ostream< \\_CharT, \\_Traits >](#) & \_\_o, size\_t \_\_n)
- `bool __Rope_is_simple` (wchar\_t \*)
- `bool __Rope_is_simple` (char \*)
- `template<class _CharT >`  
`bool __Rope_is_simple` (\_CharT \*)

- `template<class _Rope_iterator >`  
`void _Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `void _S_cond_store_eos (char &__c)`
- `template<class _CharT >`  
`void _S_cond_store_eos (_CharT &)`
- `template<class _CharT >`  
`_CharT _S_eos (_CharT *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `bool _S_is_basic_char_type (char *)`
- `template<class _CharT >`  
`bool _S_is_basic_char_type (_CharT *)`
- `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::multiplies< _Tp >)`
- `template<class _Tp >`  
`_Tp identity\_element (std::plus< _Tp >)`



- `static __Atomic_word int __val` `if` (`__gthread_active_p()`) `return` `__exchange_`  
`and_add(__mem`
- `template<typename _ForwardIter, typename _Tp >`  
`void` `iota` (`_ForwardIter __first`, `_ForwardIter __last`, `_Tp __value`)
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool` `is_heap` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_`  
`StrictWeakOrdering __comp`)
- `template<typename _RandomAccessIterator >`  
`bool` `is_heap` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`)
- `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<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 _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, typename _Cond >`  
`bool` `operator!=` (`const throw_allocator_base< _Tp, _Cond > &`, `const throw_`  
`allocator_base< _Tp, _Cond > &`)
- `template<typename _Tp >`  
`bool` `operator!=` (`const __pool_alloc< _Tp > &`, `const __pool_alloc< _Tp >`  
`&`)
- `template<typename _Tp >`  
`bool` `operator!=` (`const _Pointer_adapter< _Tp > &__lhs`, `const _Pointer_`  
`adapter< _Tp > &__rhs`)
- `template<typename _Tp >`  
`bool` `operator!=` (`int __lhs`, `const _Pointer_adapter< _Tp > &__rhs`)

- `template<typename _Tp >`  
`bool operator!= (const \_Pointer\_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool operator!= (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _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, _Tp2 __rhs)`
- `template<typename _Tp >`  
`bool operator!= (const new\_allocator< _Tp > &, const new\_allocator< _Tp > &)`
- `template<typename _Tp , typename _Poolp >`  
`bool operator!= (const \_\_mt\_alloc< _Tp, _Poolp > &, const \_\_mt\_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator!= (const malloc\_allocator< _Tp > &, const malloc\_allocator< _Tp > &)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool operator!= (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp , typename _Array >`  
`bool operator!= (const array\_allocator< _Tp, _Array > &, const array\_allocator< _Tp, _Array > &)`
- `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<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 _Tp , class _Alloc >`  
`bool operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `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< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `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<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<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<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<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 _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, _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, const _CharT *__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 _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 __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 _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< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left,`  
`_CharT __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,`  
`const rope< _CharT, _Alloc > &__right)`
- `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_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT,`  
`_Alloc > &__x, ptrdiff_t __n)`
- `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_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_`  
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _`  
`CharT __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,`  
`const rope< _CharT, _Alloc > &__right)`
- `template<typename _Cond >`  
`throw\_value\_base< _Cond > operator- (const throw\_value\_base< _Cond >`  
`&__a, const throw\_value\_base< _Cond > &__b)`
- `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<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<class _CharT, class _Alloc >`  
`ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_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_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `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 _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 _Cond >`  
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _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, _Tp2 __rhs)`
- `template<typename V, typename I, typename S >`  
`bool operator< (const character< V, I, S > &lhs, const character< V, I, S > &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<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 > &__left, const rope< _CharT, _Alloc > &__right)`
- `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)`
- `std::ostream & operator<< (std::ostream &os, const annotate\_base &__b)`
- `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)`
- `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 _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<= (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _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, _Tp2 __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<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<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 _CharT, typename _Traits, typename _Alloc, template< 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 > class _Base>`  
`bool operator== (const _CharT *__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 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 > class _Base>`  
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_`  
`__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp, typename _Cond >`  
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_`  
`allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond >`  
`bool operator== (const throw_value_base< _Cond > &__a, const throw_`  
`value_base< _Cond > &__b)`
- `template<typename _Tp >`  
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp >`  
`&)`
- `template<typename _Tp >`  
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_`  
`adapter< _Tp > &__rhs)`

- `template<typename _Tp >`  
`bool operator==(int __lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool operator==(const \_Pointer\_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator==(const \_Pointer\_adapter< _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<typename _Tp1, typename _Tp2 >`  
`bool operator==(const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename V, typename I, typename S >`  
`bool operator==(const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<typename _Tp >`  
`bool operator==(const new\_allocator< _Tp > &, const new\_allocator< _Tp > &)`
- `template<typename _Tp, typename _Poolp >`  
`bool operator==(const \_\_mt\_alloc< _Tp, _Poolp > &, const \_\_mt\_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator==(const malloc\_allocator< _Tp > &, const malloc\_allocator< _Tp > &)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator==(const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp, typename _Array >`  
`bool operator==(const array\_allocator< _Tp, _Array > &, const array\_allocator< _Tp, _Array > &)`
- `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<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 _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 > &__left, const rope< _CharT, _Alloc > &__right)`



- `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<class _CharT, class _Alloc >`  
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `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<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<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<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 _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 _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> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _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, _Tp2 __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<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<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 _CharT, typename _Traits, typename _Alloc, template< 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 > 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 > 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>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_`  
`adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_`  
`adapter< _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, _Tp2 __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<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<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 _Tp, typename _Integer >`  
`_Tp power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _`  
`RandomNumberGenerator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last,`  
`_RandomNumberGenerator &__rand)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `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<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<typename \_Cond >  
void **swap** ([throw\\_value\\_base](#)< \_Cond > &\_\_a, [throw\\_value\\_base](#)< \_Cond > &\_\_b)
- template<typename \_Tp >  
void **swap** ([\\_ExtPtr\\_allocator](#)< \_Tp > &\_\_larg, [\\_ExtPtr\\_allocator](#)< \_Tp > &\_\_rarg)
- 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 \_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)
- template<class \_CharT, class \_Alloc >  
void **swap** ([\\_Rope\\_char\\_ref\\_proxy](#)< \_CharT, \_Alloc > \_\_a, [\\_Rope\\_char\\_ref\\_proxy](#)< \_CharT, \_Alloc > \_\_b)
- 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<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 \_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 \_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)
- [\\_\\_vstring to\\_string](#) (long double \_\_val)
- [\\_\\_vstring to\\_string](#) (double \_\_val)
- [\\_\\_vstring to\\_string](#) (float \_\_val)
- [\\_\\_vstring to\\_string](#) (unsigned long long \_\_val)
- [\\_\\_vstring to\\_string](#) (long long \_\_val)
- [\\_\\_vstring to\\_string](#) (unsigned long \_\_val)
- [\\_\\_vstring to\\_string](#) (long \_\_val)

- `__vstring to_string` (unsigned `__val`)
- `__vstring to_string` (int `__val`)
- `__wvstring to_wstring` (long double `__val`)
- `__wvstring to_wstring` (double `__val`)
- `__wvstring to_wstring` (float `__val`)
- `__wvstring to_wstring` (unsigned long long `__val`)
- `__wvstring to_wstring` (long long `__val`)
- `__wvstring to_wstring` (unsigned long `__val`)
- `__wvstring to_wstring` (long `__val`)
- `__wvstring to_wstring` (unsigned `__val`)
- `__wvstring to_wstring` (int `__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` (`_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 89 of file `cast.h`.

References `__static_pointer_cast()`.

**4.1.2.2** `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 61 of file `cast.h`.

References `__static_pointer_cast()`.

Referenced by `__static_pointer_cast()`.

**4.1.2.3** `size_t __gnu_cxx::Bit_scan_forward (size_t __num) [inline]`

Generic Version of the `bsf` instruction.

Definition at line 513 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 2189 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 _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 2176 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 __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 2163 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, _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 239 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()`.

**4.1.2.8** `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) [inline]`

Concatenate string and C string.

**Parameters:**

`__lhs` First string.

`__rhs` Last string.

**Returns:**

New string with `__lhs` followed by `__rhs`.

Definition at line 222 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) [inline]`

Concatenate [character](#) and string.

**Parameters:**

`__lhs` First string.

`__rhs` Last string.

**Returns:**

New string with `__lhs` followed by `__rhs`.

Definition at line 209 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 _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

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 192 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, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Concatenate two strings.

**Parameters:**

`__lhs` First string.

`__rhs` Last string.

**Returns:**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 179 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()`.

**4.1.2.12** `template<typename _CharT, typename _Traits, typename _Alloc, typename _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 2229 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.13** `template<typename _CharT, typename _Traits, typename _Alloc, typename _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 2216 of file `vstring.h`.

**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 2203 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 _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 2309 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.16** `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 2296 of file `vstring.h`.

**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 2283 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 _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 2149 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 2136 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 __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 2112 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 529 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 2269 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 _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 2256 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 __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 2243 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 _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 2349 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 _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 2336 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 __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 2323 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 2363 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_ropes_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

### Functions

- void `__bit_allocate` (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- void `__bit_free` (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Compare` >  
`__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) [inline]`

Apply all [typelist](#) types to generator functor.

## 4.4 `__gnu_debug` Namespace Reference

GNU debug classes for public use.

### Classes

- struct `__is_same`
- class `__After_nth_from`
- 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 }

```

## Functions

- `template<typename _Iterator, typename _Sequence >`  
`bool __check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence >`  
`&__x)`
- `template<typename _Tp >`  
`bool __check_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator >`  
`bool __check_dereferenceable (_Iterator &)`
- `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_lower (_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 _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__value)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Tp >`  
`bool __check_singular (const _Tp *__ptr)`
- `template<typename _Iterator >`  
`bool __check_singular (_Iterator &__x)`
- `bool __check_singular_aux (const _Safe_iterator_base *__x)`
- `bool __check_singular_aux (const void *)`

- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__-`  
`last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__-`  
`last)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _-`  
`Predicate __pred, 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 >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &,`  
`std::input\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1`  
`&__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1`  
`&__last, const _InputIterator2 &)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &,`  
`_Predicate, std::\_\_false\_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 >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _-`  
`InputIterator &__last, std::\_\_true\_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 _Iterator, typename _Sequence >`  
`bool \_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence > &__first,`  
`const \_Safe\_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range (const \_InputIterator &__first, const \_InputIterator &__last)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux (const \_InputIterator &__first, const \_InputIterator &_`  
`__last, std::__false_type)`
- `template<typename _Integral >`  
`bool \_\_valid\_range\_aux (const \_Integral &, const \_Integral &, std::__true_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)`
- `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 _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 \_CharT *__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 _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 _CharT *__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 _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 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 _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 _CharT *__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 _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 _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 _CharT *__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 _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 _CharT *__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 _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 _CharT *__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 _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 _CharT *__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 _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, 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()`.

**4.4.2.2** `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.

**4.4.2.3** `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.

**4.4.2.4** `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.

**4.4.2.5** `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.

**4.4.2.6** `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 48 of file safe\_iterator.h.

**4.4.2.7** `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.

**4.4.2.8** `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.

```
4.4.2.9 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`.

```
4.4.2.10 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 _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()`.

```
4.4.2.12 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_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.

**4.4.2.14** `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 [\\_PMWMSSortingData](#)  
*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.*
- class [\\_VoidFunctor](#)  
*Functor that does nothing.*
- 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 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 `_Iter`, typename `_OutputIterator`>  
`_OutputIterator` `__copy_tail` (`std::pair`< `_Iter`, `_Iter` > `__b`, `std::pair`< `_Iter`, `_Iter` > `__e`, `_OutputIterator` `__r`)
- void `__decode2` (`_CASable` `__x`, int &`__a`, int &`__b`)
- template<typename `_RAIter`, typename `_DifferenceTp`>  
void `__determine_samples` (`_PMWMSortingData`< `_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`, `constant_size_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`, `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`, `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`)
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result > _UserOp __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`)
- `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`, `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` (`_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 _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 _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 _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, typename [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, [default\\_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, [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 >`  
`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,`  
`multiway\_mergesort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_-`  
`Parallelism __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 _IIter, class _OutputIterator >`  
`_OutputIterator \_\_parallel\_unique\_copy (_IIter __first, _IIter __last, \_-`  
`OutputIterator __result)`
- `template<typename _IIter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator \_\_parallel\_unique\_copy (_IIter __first, _IIter __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, type-`  
`name _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 multiseq\_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, 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::sampling\_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::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::sequential\_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, 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, 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, sampling\_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, 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, [\\_\\_gnu\\_parallel::sequential\\_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, [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, [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, [\\_\\_gnu\\_parallel::exact\\_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\_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, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _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, type-`  
`name _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, type-`  
`name _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

## 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) [inline]`

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 335 of file `parallel/compatibility.h`.

References `__compare_and_swap_32()`, and `__compare_and_swap_64()`.

Referenced by `__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 239 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 282 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 (_PMWMSortingData< _RAIter > * __sd, _DifferenceTp __num_samples) [inline]`

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::PMWMSortingData<_RAIter>::_M_samples`, `__gnu_parallel::PMWMSortingData<_RAIter>::_M_source`, `__gnu_parallel::PMWMSortingData<_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 185 of file `parallel/compatibility.h`.

References `__fetch_and_add_32()`, and `__fetch_and_add_64()`.

Referenced by `__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, constant_size_blocks_tag) [inline]`

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`, and `__gnu_parallel::_Settings::find_sequential_search_size`.

**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, growing_blocks_tag) [inline]`

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\\_initial\\_block\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_maximum\\_block\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_increasing\\_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 187 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_increasing_factor`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_maximum_block_size`, and `__gnu_parallel::_Settings::find_sequential_search_size`.

**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, equal_split_tag) [inline]`

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()`.

**4.6.4.14** `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) [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`.

**4.6.4.15** `template<typename _Iiter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::__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) [inline]`

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) [inline]

```

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) [inline]

```

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) [inline]`

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) [inline]`

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`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.20** `template<typename _IIter, typename _Compare > bool  
__gnu_parallel::__is_sorted(_IIter __begin, _IIter __end, _Compare  
__comp) [inline]`

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) [inline]`

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 433 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) [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. 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) [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.

**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 _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 `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.26** `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()`.



**4.6.4.27** `template<typename _RAIter, typename _Compare > void  
__gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter  
__nth, _RAIter __end, _Compare __comp) [inline]`

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 336 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, `__gnu_parallel::_Settings::partition_minimal_n`, and `std::swap()`.

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) [inline]`

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 426 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) [inline]`

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 199 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, and `_GLIBCXX_CALL`.

**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)  
[inline]`

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)
[inline]
```

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()`, `std::accumulate()`, `equally_split()`, 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)
[inline]
```

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 `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::partition_chunk_share`, `__gnu_parallel::_Settings::partition_chunk_size`, and `std::swap()`.

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 =
_RandomNumberGenerator()) [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 511 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) [inline]
```

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)  
[inline]`

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.



**4.6.4.38** `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.

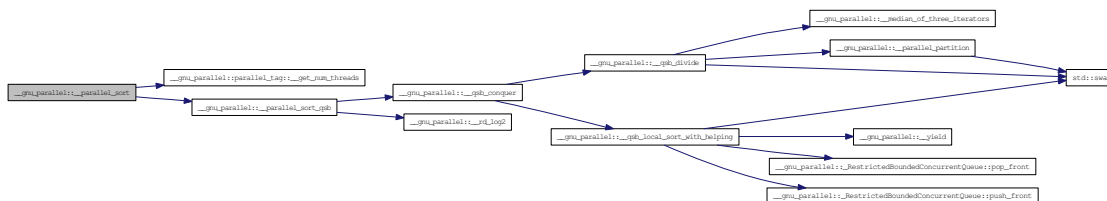
**Parameters:**

- `__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_qsb()`, 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, quicksort_tag __parallelism) [inline]`

Choose quicksort for parallel sorting.

**Parameters:**

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 136 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:



**4.6.4.40** `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.

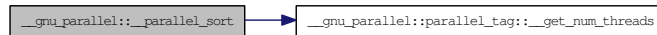
**Parameters:**

- `__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:



**4.6.4.41** `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:**

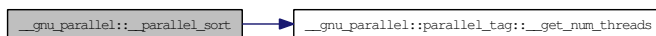
- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__comp` Comparator.



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:



**4.6.4.42** `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.

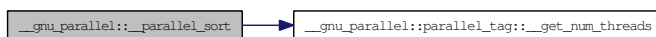
`__end` End iterator of input sequence.

`__comp` Comparator.

Definition at line 74 of file sort.h.

References \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads(), 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) [inline]`

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)
 [inline]
```

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()`.

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) [inline]
```

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)
 [inline]
```

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()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

```
4.6.4.47 template<typename _IIter , class _OutputIterator > _OutputIterator
 __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __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) [inline]`

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) [inline]`

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) [inline]`

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()`, `__parallel_partition()`, and `std::swap()`.

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) [inline]`

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::_RestrictedBoundedConcurrentQueue<_Tp >::pop_front()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp >::push_front()`, and `std::swap()`.

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) [inline]`

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) [inline]`

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) [inline]`

Sequential multi-way merging switch. The `__GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime 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)  
[inline]`

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 402 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) [inline]`

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 _IIter > void __gnu_parallel::__shrink_and_double (std::vector<_IIter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice) [inline]`

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 352 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) [inline]`

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)  
 [inline]`

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 _Iiter , typename _FunctorType > size_t  
 __gnu_parallel::list_partition (const _Iiter __begin, const _Iiter  
 __end, _Iiter * __starts, size_t * __lengths, const int __num_parts,  
 _FunctorType & __f, int __oversampling = 0) [inline]`

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  $\{ \{ \text{__end} \} - \{ \text{__begin} \} \} \text{__elements}$ . 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> ())
[inline]

```

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()`, `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> ())
[inline]

```

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()`, `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)  
[inline]`

Multway 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) [inline]`

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) [inline]`

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)  
[inline]`

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) [inline]`

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) [inline]`

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

**Parameters:**

- \_\_stable* The value must the same as for the used LoserTrees.
- UnguardedLoserTree* \_Loser Tree variant to use for the unguarded merging.
- GuardedLoserTree* \_Loser Tree 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) [inline]`

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)
 [inline]
```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1032 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`.

```
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) [inline]
```

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; __j < 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) [inline]`

Parallel multi-way merge routine. The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime 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`, and `_GLIBCXX_PARALLEL_LENGTH`.

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) [inline]`

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`, and `std::vector< _Tp, _Alloc >::resize()`.

**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) [inline]`

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`, 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::std::vector< \_\_cost\_factor \* > **\_\_cost\_factor\_vector**
- typedef void \* **\_\_instruction\_address\_t**
- typedef pthread\_mutex\_t **\_\_mutex\_t**
- typedef const void \* **\_\_object\_t**
- typedef std::std::vector< \_\_instruction\_address\_t > **\_\_stack\_npt**
- typedef \_\_stack\_npt \* **\_\_stack\_t**
- typedef std::std::vector< [\\_\\_warning\\_data](#) > **\_\_warning\_vector\_t**

## Enumerations

- enum **\_\_state\_type** { **\_\_ON**, **\_\_OFF**, **\_\_INVALID** }

## Functions

- `size_t __env_to_size_t` (const char \* \_\_env\_var, size\_t \_\_default\_value)
- `__cost_factor_vector * & __get__cost_factors` ()
- `__mutex_t & __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` ()
- `__state_type & __get__state` ()
- `__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` ()
- `size_t & __get__S_max_mem` ()
- `size_t & __get__S_max_stack_depth` ()
- `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` ()
- `bool __is_invalid` ()
- `bool __is_off` ()
- `bool __is_on` ()
- `void __lock` (\_\_mutex\_t & \_\_m)
- `int __log2` (size\_t \_\_size)
- `int __log_magnitude` (float f)
- `float __map_erase_cost` (size\_t \_\_size)
- `float __map_find_cost` (size\_t \_\_size)
- `float __map_insert_cost` (size\_t \_\_size)
- `size_t __max` (size\_t \_\_a, size\_t \_\_b)
- `size_t __max_mem` ()

- `size_t __min` (`size_t __a`, `size_t __b`)
- `FILE * __open_output_file` (`const char *extension`)
- `bool __profexx_init` (`void`)
- `void __profexx_init_unconditional` ()
- `void __read_cost_factors` ()
- `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` ()
- `__size` (`const __stack_t &__stack`)
- `size_t __stack_max_depth` ()
- `void __trace_hash_func_construct` (`const void *`)
- `void __trace_hash_func_destruct` (`const void *`, `size_t`, `size_t`, `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 *`, `size_t`)
- `void __trace_hashtable_size_destruct` (`const void *`, `size_t`, `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 *`, `size_t`, `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 *`, `size_t`)
- `void __trace_list_to_set_insert` (`const void *`, `size_t`, `size_t`)
- `void __trace_list_to_set_invalid_operator` (`const void *`)
- `void __trace_list_to_set_iterate` (`const void *`, `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 *`, `size_t`, `size_t`)
- `void __trace_list_to_vector_invalid_operator` (`const void *`)
- `void __trace_list_to_vector_iterate` (`const void *`, `size_t`)

- void `__trace_list_to_vector_report` (FILE \*`__f`, `__warning_vector_t` &`__warnings`)
- void `__trace_list_to_vector_resize` (const void \*, `size_t`, `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 \*, `size_t`, `size_t`)
- void `__trace_map_to_unordered_map_find` (const void \*, `size_t`)
- void `__trace_map_to_unordered_map_init` ()
- void `__trace_map_to_unordered_map_insert` (const void \*, `size_t`, `size_t`)
- void `__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__trace_map_to_unordered_map_iterate` (const void \*, `size_t`)
- void `__trace_map_to_unordered_map_report` (FILE \*`__f`, `__warning_vector_t` &`__warnings`)
- void `__trace_vector_size_construct` (const void \*, `size_t`)
- void `__trace_vector_size_destruct` (const void \*, `size_t`, `size_t`)
- void `__trace_vector_size_init` ()
- void `__trace_vector_size_report` (FILE \*, `__warning_vector_t` &)
- void `__trace_vector_size_resize` (const void \*, `size_t`, `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 \*, `size_t`)
- void `__trace_vector_to_list_init` ()
- void `__trace_vector_to_list_insert` (const void \*, `size_t`, `size_t`)
- void `__trace_vector_to_list_invalid_operator` (const void \*)
- void `__trace_vector_to_list_iterate` (const void \*, `size_t`)
- void `__trace_vector_to_list_report` (FILE \*, `__warning_vector_t` &)
- void `__trace_vector_to_list_resize` (const void \*, `size_t`, `size_t`)
- bool `__turn` (`__state_type` `__s`)
- bool `__turn_off` ()
- bool `__turn_on` ()
- void `__unlock` (`__mutex_t` &`__m`)
- void `__write` (FILE \*`__f`, const `__stack_t` `__stack`)
- void `__write_cost_factors` ()

### 4.8.1 Detailed Description

GNU profile code for public use.

## 4.8.2 Function Documentation

### 4.8.2.1 `__mutex_t& __gnu_profile::__get__global_lock ()` [`inline`]

Pthread mutex wrapper.

Definition at line 82 of file `profiler_trace.h`.

### 4.8.2.2 `bool __gnu_profile::__profctx_init (void)` [`inline`]

This function must be called by each instrumentation point. The common path is inlined fully.

Definition at line 672 of file `profiler_trace.h`.

### 4.8.2.3 `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 478 of file `profiler_trace.h`.



## 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](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

- 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.*
- 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.*
- class [\\_Has\\_result\\_type\\_helper](#)
- 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 [\\_List\\_node\\_base](#)  
Common part of a 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 `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< void \\* >](#)  
*Explicit specialization for void\*.*
- struct [atomic< wchar\\_t >](#)  
*Explicit specialization for wchar\_t.*
- 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\\_iostream](#)

*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\\_istream](#)

*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\\_ostream](#)

*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\\_assign](#)  
*[has\\_nothrow\\_assign](#)*
- struct [has\\_nothrow\\_copy\\_constructor](#)  
*[has\\_nothrow\\_copy\\_constructor](#)*
- struct [has\\_nothrow\\_default\\_constructor](#)  
*[has\\_nothrow\\_default\\_constructor](#)*
- struct [has\\_trivial\\_assign](#)  
*[has\\_trivial\\_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< ::bitset< _Nb > >`  
*std::hash specialization for `bitset`.*
- struct `hash< ::vector< bool, _Alloc > >`  
*std::hash specialization for `vector<bool>`.*
- 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< _Tp * >`  
*Partial specializations for pointer types.*
- struct `hash< error_code >`  
*std::hash specialization for `error_code`.*
- struct `hash< string >`  
*std::hash specialization for `string`.*
- struct `hash< thread::id >`  
*std::hash specialization for `thread::id`.*
- struct `hash< u16string >`  
*std::hash specialization for `u16string`.*
- struct `hash< u32string >`

*std::hash* specialization for *u32string*.

- struct [hash< wstring >](#)  
*std::hash* specialization for *wstring*.
- struct [identity](#)  
*identity*
- 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 sub-expression 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\\_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_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_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`  
*Traits class for iterators.*
- 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 `num_punct`  
*Primary class template `num_punct`.  
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 `num_punct` facet is used by streams for many I/O operations involving numbers.*
- class `num_punct_byname`  
*class `num_punct_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`  
*`pair` holds two objects of arbitrary type.*
- class `piecewise_constant_distribution`  
*A `piecewise_constant_distribution` random number distribution.*
- 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\\_greater](#)  
*[ratio\\_greater](#)*
- struct [ratio\\_greater\\_equal](#)  
*[ratio\\_greater\\_equal](#)*
- struct [ratio\\_less](#)

*ratio\_less*

- struct `ratio_less_equal`  
*ratio\_less\_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, _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*
- 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[ ], _Tp_Deleter >`  
*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.*

- 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 struct std::\_\_atomic\_flag\_base **\_\_atomic\_flag\_base**
- typedef FILE **\_\_c\_file**
- typedef \_\_locale\_t **\_\_c\_locale**
- typedef \_\_gthread\_mutex\_t **\_\_c\_lock**
- typedef unsigned long **\_Bit\_type**
- typedef [atomic\\_short](#) **atomic\_int\_fast16\_t**
- typedef [atomic\\_int](#) **atomic\_int\_fast32\_t**
- typedef [atomic\\_llong](#) **atomic\_int\_fast64\_t**
- typedef [atomic\\_schar](#) **atomic\_int\_fast8\_t**
- typedef [atomic\\_short](#) **atomic\_int\_least16\_t**
- typedef [atomic\\_int](#) **atomic\_int\_least32\_t**
- typedef [atomic\\_llong](#) **atomic\_int\_least64\_t**
- typedef [atomic\\_schar](#) **atomic\_int\_least8\_t**
- typedef [atomic\\_llong](#) **atomic\_intmax\_t**
- typedef [atomic\\_long](#) **atomic\_intptr\_t**
- typedef [atomic\\_long](#) **atomic\_ptrdiff\_t**

- typedef [atomic\\_ulong](#) **atomic\_size\_t**
- typedef [atomic\\_long](#) **atomic\_ssize\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_fast16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_fast32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_fast64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_fast8\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_least16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_least32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_least64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_least8\_t**
- typedef [atomic\\_ullong](#) **atomic\_uintmax\_t**
- typedef [atomic\\_ulong](#) **atomic\_uintptr\_t**
- typedef [ratio](#)< 1, 1000000000000000000 > **atto**
- typedef [ratio](#)< 1, 100 > **centi**
- 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 [ratio](#)< 10, 1 > **deca**
- typedef [ratio](#)< 1, 10 > **deci**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- typedef [ratio](#)< 1000000000000000000, 1 > **exa**
- typedef [integral\\_constant](#)< bool, false > **false\_type**
- typedef [ratio](#)< 1, 1000000000000000000 > **femto**
- typedef [basic\\_filebuf](#)< char > **filebuf**
- typedef [basic\\_fstream](#)< char > **fstream**
- typedef [ratio](#)< 1000000000, 1 > **giga**
- typedef [ratio](#)< 100, 1 > **hecto**
- 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 [basic\\_istream](#)< char > **istream**
- typedef [ratio](#)< 1000, 1 > **kilo**
- typedef [shuffle\\_order\\_engine](#)< [minstd\\_rand0](#), 256 > **knuth\_b**
- typedef [ratio](#)< 1000000, 1 > **mega**
- typedef enum [std::memory\\_order](#) **memory\_order**
- typedef [ratio](#)< 1, 1000000 > **micro**
- typedef [ratio](#)< 1, 1000 > **milli**
- 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 [ratio](#)< 1, 1000000000 > **nano**
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > **ofstream**
- typedef [basic\\_ostream](#)< char > **ostream**
- typedef [basic\\_ostringstream](#)< char > **ostringstream**
- typedef [ratio](#)< 1000000000000000, 1 > **peta**
- typedef [ratio](#)< 1, 1000000000000 > **pico**
- 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 [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 **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 [ratio](#)< 1000000000000, 1 > **tera**
- 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 \* > **wcsmatch**
- typedef [regex\\_iterator](#)< const wchar\_t \* > **wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > **wcregex\_token\_iterator**

- typedef [sub\\_match](#)< const wchar\_t \* > [wsub\\_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 > [wsub\\_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 `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 _RandomAccessIterator , typename _Distance , typename _Tp , typename _Compare >`  
`void __adjust_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`,  
`_Distance __len`, `_Tp __value`, `_Compare __comp`)
- `template<typename _RandomAccessIterator , typename _Distance , typename _Tp >`  
`void __adjust_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`,  
`_Distance __len`, `_Tp __value`)
- `template<typename _RandomAccessIterator , typename _Distance >`  
`void __advance` (`_RandomAccessIterator & __i`, `_Distance __n`, `random_access_iterator_tag`)
- `template<typename _BidirectionalIterator , typename _Distance >`  
`void __advance` (`_BidirectionalIterator & __i`, `_Distance __n`, `bidirectional_iterator_tag`)
- `template<typename _InputIterator , typename _Distance >`  
`void __advance` (`_InputIterator & __i`, `_Distance __n`, `input_iterator_tag`)
- `void __atomic_flag_wait_explicit` (`_atomic_flag_base *`, `memory_order`)  
`throw ()`
- `__attribute__((__pure__))` `bool __verify_grouping`(`const char * __grouping`)
- `__attribute__((__const__))` `__atomic_flag_base * __atomic_flag_for_address`(`const void * __z`) `throw ()`
- `memory_order __calculate_memory_order` (`memory_order __m`)
- `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_function (_Member _Class::*&_`  
`_p)`
- `template<typename _Functor >`  
`_Functor & __callable_function (_Functor &_f)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *_f)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
  
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _Tp >`  
`_Tp __cmath_power (_Tp, unsigned int)`
- `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 com-`  
`plex< _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<>`  
`void __convert_to_v (const char *, long double &, 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 *, float &, ios_base::iostate &, const __c_`  
`locale &) throw ()`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, 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, _CharT * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_`  
`iterator< _CharT > __last, _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, ostreambuf_`  
`iterator< _CharT > >::__type __copy_move_a2 (_CharT * __first, _CharT * _`  
`__last, ostreambuf_iterator< _CharT > __result)`
- `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, _CharT * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT >`  
`>, istreambuf_iterator< _CharT, char_traits< _CharT > >, _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, ostreambuf_`  
`iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (_`  
`CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `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 _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _`  
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator _`  
`_result, input_iterator_tag)`
- `template streamsize __copy_streambufs (basic_streambuf< wchar_t > *,`  
`basic_streambuf< wchar_t > *)`
- `template streamsize __copy_streambufs (basic_streambuf< char > *, basic_`  
`streambuf< char > *)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin,`  
`basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin,`  
`basic_streambuf< wchar_t > *__sbout, bool &__ineof)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_`  
`streambuf< char > *__sbout, bool &__ineof)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *,`  
`basic_streambuf< _CharT, _Traits > *, bool &)`
- `size_t __deque_buf_size (size_t __size)`
- `template<typename _RandomAccessIterator >`  
`iterator_traits< _RandomAccessIterator >::difference_type __distance (_`  
`RandomAccessIterator __first, _RandomAccessIterator __last, random_access_`  
`iterator_tag)`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type __distance (_InputIterator _`  
`_first, _InputIterator __last, input_iterator_tag)`

- `template<typename _II1, typename _II2 >`  
`bool __equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type __fill_a`  
`( _Tp * __first, _Tp * __last, const _Tp & __c)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, void >::__type __-`  
`fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `void __fill_bvector (_Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `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 _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 _RandomAccessIterator, typename _Compare >`  
`void __final_insertion_sort (_RandomAccessIterator __first, __-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void __final_insertion_sort (_RandomAccessIterator __first, __-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator __find (_RandomAccessIterator __first, __-`  
`RandomAccessIterator __last, const _Tp & __val, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator __find (_InputIterator __first, _InputIterator __last, const _Tp & __-`  
`__val, input\_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 _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,`  
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, __-`  
`BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_-`  
`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 _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_`  
`iterator\_tag, forward\_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 (_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 _InputIterator, typename _Predicate >`  
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _`  
`Predicate __pred, input\_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_adapter (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, typename _Compare >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator >`  
`void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `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, typename _Compare >`  
`void \_\_introsort (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introsort (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__nth, _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 _Size >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Size __depth_limit)`
- `template<typename _Functor, typename... _Args>`  
`enable\_if< (is\_pointer< _Functor >::value &&is\_function< typename`  
`remove\_pointer< _Functor >::type >::value), typename result_of< _-`  
`Functor(_Args...)>::type >::type \_\_invoke (_Functor __f, _Args &&...__args)`
- `template<typename _Functor, typename... _Args>`  
`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)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance`  
`__n)`

- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _-`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `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)`
- `long long __lg (long long __n)`
- `long __lg (long __n)`
- `int __lg (int __n)`
- `template<typename _Size >`  
`_Size __lg (_Size __n)`
- `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 _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 _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3, typename _Compare >`  
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare`  
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3 >`  
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `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, typename _Compare >`
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
 - `template<typename _RandomAccessIterator, typename _Pointer >`
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer)`
 - `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 _BidirectionalIterator, typename _Distance >`
`void __merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2)`
 - `template<typename _Iterator >`
`_Miter_base< _Iterator >::iterator_type __miter_base (_Iterator __it)`
 - `template<typename _Iterator, typename _Compare >`
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare`
`__comp)`
 - `template<typename _Iterator >`
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
 - `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 wostream & __ostream_insert (wostream &, const wchar_t *, stream-`
`size)`
 - `template ostream & __ostream_insert (ostream &, const char *, streamsize)`
 - `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _-`
`CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
 - `template<typename _CharT, typename _Traits >`
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _-`
`CharT *__s, streamsize __n)`
 - `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
 - `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last,`
`_Predicate __pred, forward_iterator_tag)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _RandomAccessIterator __result)`
- `template<typename _Tp >`
`_Tp __pow_helper (_Tp __x, int __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`
`Compare >`
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _`
`Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _`
`Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`random_access_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _`
`BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator >`
`void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _`
`ForwardIterator __last, forward_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 _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)`
- `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 >`
`_RandomAccessIter __search_n (_RandomAccessIter __first, _-`
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_`
`access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last,`
`_Integer __count, const _Tp &__val, std::forward_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, typename`
`_Compare >`
`void __stable_sort_adaptive (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`
`_Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`
`void __stable_sort_adaptive (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`

- `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_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 &&,...) __attribute__((__noreturn__))`

- `template<typename _Ex >`
`void __throw_with_nested (_Ex &&, const nested_exception *=0) __-`
`attribute__((__noreturn__))`
- `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...> &, 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...> &, 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...> &, const tuple<`
`_UElements...> &&_u, const __index_holder< _UIdx...> &)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `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, typename _Compare >`
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __-`
`first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __-`
`first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `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_range (_ForwardIterator __first, _-`
`ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `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 _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_RandomAccessIterator __first, _-`
`Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_InputIterator __first, _Size __n, _-`
`ForwardIterator __result, input_iterator_tag)`
- `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 _Allocator >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x, _Allocator &__alloc)`
- `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 _Tp2 >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`
`&__x, allocator< _Tp2 > &)`
- `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 _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 _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last,`
`_ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`
`forward_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 >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __`
`__last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `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 > __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< size_t > __i)`
- `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)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __k)`

- `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 > __a, _Array< bool > __m, _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, size_t __n, _Array< _Tp > __b,`
`_Array< size_t > __i)`
- `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 > __a, size_t __n, size_t __s1, _Array<`
`_Tp > __b, size_t __s2)`
- `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 __s, _Array<`
`_Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `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 (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b, const size_t *__restrict __i)`
- `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 __src, size_t __n, size_t __s1,`
`_Tp *__restrict __dst, size_t __s2)`
- `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 __a, size_t __n, size_t __s, _Tp`
`*__restrict __b)`

- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict__ __a, size_t __n, _Tp *__-`
`restrict__ __b)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m,`
`_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, 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 >`
`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 *__restrict__ __a, size_t __n,`
`size_t __s, _Tp *__restrict__ __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__-`
`restrict__ __o)`
- `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 (_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 (_Array< _Tp > __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 (_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, size_t __s, const _Tp`
`&__t)`

- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, 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)`
- `template<typename _CharT, typename _OutIter >`
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s,`
`const _CharT *__ws, int __len)`
- `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, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `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<`
`size_t > __i, 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 >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `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, _Array< _Tp`
`> __b, size_t __n, size_t __s)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`
`size_t __s, _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, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `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, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `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 >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `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, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _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, _-`
`Array< _Tp > __b)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `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 >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `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, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `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< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `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, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_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, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `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, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `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, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _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, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `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, _Array< _Tp > __b, _Array< bool > __m)`
- `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< size_t > __i, const _Expr< _Dom, _Tp > &__e, 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 >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, 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< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t _`
`__s, _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 >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp`
`&__t)`
- `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, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `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< size_t >`
`__i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `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, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `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 >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`

- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _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, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `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< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, 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, _Array< _Tp > __b, _Array< size_t > __i)`
- `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, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `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, const`
`_Tp &__t)`
- `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, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`
- `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< size_t > __i,`
`const _Expr< _Dom, _Tp > &__e, 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__plus (_Array< _Tp > __a, _Array< size_t > __i,`
`_Array< _Tp > __b, size_t __n)`
- `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 >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s,`
`_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, _Array<`
`_Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp`
`&__t)`
- `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, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`

- `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< size_t >`
`__i, 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 >`
`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_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, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t`
`__s, _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, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `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 >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `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, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp >`
`void _Array_augmented__shift_right (_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, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `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, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _T1, typename _T2 >`
`void _Construct (_T1 *__p, _T2 &&__value)`
- `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)`
- `template<typename _ForwardIterator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp >`
`void _Destroy (_Tp *__pointer)`
- `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) _GLIBC_NORETURN throw ()`
- `template<typename _Tp >`
`_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > abs (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > abs`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type abs`
`(_Tp __x)`
- `long double abs (long double __x)`
- `float abs (float __x)`
- `double abs (double __x)`
- `template<typename _Tp >`
`_Tp abs (const complex< _Tp > &)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _-`
`BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _Tp >`
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > acos (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type >`
`acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`std::complex< _Tp > acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`acos (_Tp __x)`
- `long double acos (long double __x)`
- `float acos (float __x)`
- `template<typename _Tp >`
`std::complex< _Tp > acosh (const std::complex< _Tp > &__z)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter)`

- `template<typename _InputIterator, typename _Distance >`
`void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator, typename _Predicate >`
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp > allocate_shared (_Alloc __a, _Args &&... __args)`
- `template<typename _InputIterator, typename _Predicate >`
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `template<typename _Tp >`
`_Tp arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asin (_Tp __x)`
- `long double asin (long double __x)`
- `float asin (float __x)`
- `template<typename _Tp >`
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `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 _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...) >::__type > async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`std::complex< _Tp > atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`atan (_Tp __x)`
- `long double atan (long double __x)`
- `float atan (float __x)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `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, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `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<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, _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<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 _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<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)`
- `long double atan2 (long double __y, long double __x)`
- `float atan2 (float __y, float __x)`

- `template<typename _Tp >`
`std::complex< _Tp > atanh (const std::complex< _Tp > &_z)`
- `int atexit (void(*)()) throw ()`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong (__atomic_base< _ITp > *_a, _ITp *_i1, _ITp __i2)`
- `bool atomic_compare_exchange_strong (atomic_bool *_a, bool *_i1, bool __i2)`
- `bool atomic_compare_exchange_strong (atomic_address *_a, void **_v1, void *_v2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong_explicit (__atomic_base< _ITp > *_a, _ITp *_i1, _ITp __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 (atomic_address *_a, void **_v1, void *_v2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_weak (__atomic_base< _ITp > *_a, _ITp *_i1, _ITp __i2)`
- `bool atomic_compare_exchange_weak (atomic_bool *_a, bool *_i1, bool __i2)`
- `bool atomic_compare_exchange_weak (atomic_address *_a, void **_v1, void *_v2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_weak_explicit (__atomic_base< _ITp > *_a, _ITp *_i1, _ITp __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)`
- `bool atomic_compare_exchange_weak_explicit (atomic_address *_a, void **_v1, void *_v2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`_ITp atomic_exchange (__atomic_base< _ITp > *_a, _ITp __i)`
- `bool atomic_exchange (atomic_bool *_a, bool __i)`
- `void * atomic_exchange (atomic_address *_a, void *_v)`
- `template<typename _ITp >`
`_ITp atomic_exchange_explicit (__atomic_base< _ITp > *_a, _ITp __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)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add (__atomic_base< _ITp > *_a, _ITp __i)`

- void * **atomic_fetch_add** (atomic_address *__a, ptrdiff_t __d)
- template<typename _ITp >
_ITp **atomic_fetch_add_explicit** (__atomic_base< _ITp > *__a, _ITp __i, [memory_order](#) __m)
- void * **atomic_fetch_add_explicit** (atomic_address *__a, ptrdiff_t __d, [memory_order](#) __m)
- template<typename _ITp >
_ITp **atomic_fetch_and** (__atomic_base< _ITp > *__a, _ITp __i)
- 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_explicit** (__atomic_base< _ITp > *__a, _ITp __i, [memory_order](#) __m)
- template<typename _ITp >
_ITp **atomic_fetch_sub** (__atomic_base< _ITp > *__a, _ITp __i)
- void * **atomic_fetch_sub** (atomic_address *__a, ptrdiff_t __d)
- template<typename _ITp >
_ITp **atomic_fetch_sub_explicit** (__atomic_base< _ITp > *__a, _ITp __i, [memory_order](#) __m)
- void * **atomic_fetch_sub_explicit** (atomic_address *__a, ptrdiff_t __d, [memory_order](#) __m)
- template<typename _ITp >
_ITp **atomic_fetch_xor** (__atomic_base< _ITp > *__a, _ITp __i)
- template<typename _ITp >
_ITp **atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, _ITp __i, [memory_order](#) __m)
- void **atomic_flag_clear** (__atomic_flag_base *__a)
- void **atomic_flag_clear_explicit** (__atomic_flag_base *, [memory_order](#)) throw ()
- void **atomic_flag_clear_explicit** (atomic_flag *__a, [memory_order](#) __m)
- bool **atomic_flag_test_and_set** (__atomic_flag_base *__a)
- bool **atomic_flag_test_and_set_explicit** (__atomic_flag_base *, [memory_order](#)) throw ()
- bool **atomic_flag_test_and_set_explicit** (atomic_flag *__a, [memory_order](#) __m)
- template<typename _ITp >
bool **atomic_is_lock_free** (const __atomic_base< _ITp > *__a)
- bool **atomic_is_lock_free** (const atomic_bool *__a)
- bool **atomic_is_lock_free** (const atomic_address *__a)
- template<typename _ITp >
_ITp **atomic_load** (const __atomic_base< _ITp > *__a)

- `bool atomic_load (const atomic_bool * __a)`
- `void * atomic_load (const atomic_address * __a)`
- `template<typename _ITp >
_ITp atomic_load_explicit (const __atomic_base< _ITp > * __a, memory_order __m)`
- `bool atomic_load_explicit (const atomic_bool * __a, memory_order __m)`
- `void * atomic_load_explicit (const atomic_address * __a, memory_order __m)`
- `template<typename _ITp >
void atomic_store (__atomic_base< _ITp > * __a, _ITp __i)`
- `void atomic_store (atomic_bool * __a, bool __i)`
- `void atomic_store (atomic_address * __a, void * __v)`
- `template<typename _ITp >
void atomic_store_explicit (__atomic_base< _ITp > * __a, _ITp __i, memory_order __m)`
- `void atomic_store_explicit (atomic_bool * __a, bool __i, memory_order __m)`
- `void atomic_store_explicit (atomic_address * __a, void * __v, memory_order __m)`
- `template<typename _Container >
back_insert_iterator< _Container > back_inserter (_Container & __x)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >
bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >
bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _FIter, typename _Tp, typename _Compare >
bool binary_search (_FIter, _FIter, const _Tp &, _Compare)`
- `template<typename _FIter, typename _Tp >
bool binary_search (_FIter, _FIter, const _Tp &)`
- `template<typename _Result, typename _Functor, typename... _ArgTypes >
_Bind_result< _Result, typename Maybe_wrap_member_pointer< _Functor >::type(_ArgTypes...) > bind (_Functor __f, _ArgTypes... __args)`
- `template<typename _Functor, typename... _ArgTypes >
_Bind< typename Maybe_wrap_member_pointer< _Functor >::type(_ArgTypes...) > 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)`
- `template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil (_Tp __x)`

- long double **ceil** (long double __x)
- float **ceil** (float __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **conj** (_Tp __x)
- template<typename _Tp >
complex< _Tp > **conj** (const **complex**< _Tp > &)
- template<typename _Tp, typename _Tp1 >
shared_ptr< _Tp > **const_pointer_cast** (const **shared_ptr**< _Tp1 > &__r)
- 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 _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 _II, typename _OI >
_OI **copy** (_II __first, _II __last, _OI __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 _IIter, typename _OIter >
_OIter **copy** (_IIter, _IIter, _OIter)
- 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 _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, 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 _InputIterator, typename _OutputIterator, typename _Predicate >
_OutputIterator **copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _IIter, typename _OIter, typename _Predicate >
_OIter **copy_if** (_IIter, _IIter, _OIter, _Predicate)

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __-`
`result)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter copy_n (_Iter, _Size, _OIter)`
- `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 >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cos`
`(_Tp __x)`
- `long double cos (long double __x)`
- `float cos (float __x)`
- `template<typename _Tp >`
`complex< _Tp > cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`
`cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`cosh (_Tp __x)`
- `long double cosh (long double __x)`
- `float cosh (float __x)`
- `template<typename _Tp >`
`complex< _Tp > cosh (const complex< _Tp > &)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type count (_InputIterator __first,`
`_InputIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __-`
`first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter, _Iter, _Predicate)`
- `exception_ptr current_exception () throw ()`
- `ios_base & dec (ios_base &__base)`
- `template<typename _Tp >`
`add_rvalue_reference< _Tp >::type declval ()`

- `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 ostream & endl (ostream &)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits >`
`&__os)`
- `template ostream & ends (ostream &)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits >`
`&__os)`
- `template<typename _II1, typename _II2 >`
`bool equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`
`bool equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _BinaryPredicate`
`__binary_pred)`
- `template<typename _Iiter1, typename _Iiter2 >`
`bool equal (_Iiter1, _Iiter1, _Iiter2)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __-`
`first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __-`
`first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `void exit (int) _GLIBC_NORETURN throw ()`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > exp`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp`
`(_Tp __x)`
- `long double exp (long double __x)`
- `float exp (float __x)`
- `template<typename _Tp >`
`complex< _Tp > exp (const complex< _Tp > &)`

- `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)`
- long double **[fabs](#)** (long double __x)
- float **[fabs](#)** (float __x)
- void **[fill](#)** (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)
- `template<typename _ForwardIterator, typename _Tp >`
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `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 _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 _InputIterator, typename _Tp >`
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _IIter, typename _Tp >`
`_IIter find (_IIter, _IIter, const _Tp &)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`
`_FIter1 find_end (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _FIter1, typename _FIter2 >`
`_FIter1 find_end (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _-`
`ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate _-`
`__pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `ios_base & fixed (ios_base & __base)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >>::__value, double >::__type`
`floor (_Tp __x)`
- long double `floor` (long double __x)
- float `floor` (float __x)
- `template ostream & flush (ostream &)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits >`
`& __os)`
- long double `fmod` (long double __x, long double __y)
- float `fmod` (float __x, float __y)
- `template<typename _InputIterator, typename _Function >`
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Iter, typename _Funct >`
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`
`name std::remove_reference< _Tp >::type && __t)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`
`name std::identity< _Tp >::type __t)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`
`name std::identity< _Tp >::type && __t)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`
`name std::identity< _Tp >::type & __t)`

- `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)`
- `template<typename _ForwardIterator, typename _Generator >`
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`
`_RealType generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `_GLIBCXX_CONST const error_category & generic_category () throw ()`
- `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, class _Tp1, class _Tp2 >`
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`const _Tp & get (const array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`_Tp & get (array< _Tp, _Nm > &__arr)`
- `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 __i, typename... _Elements>`
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type get (tuple< _Elements...> &__t)`
- `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, 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 _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 basic_istream< wchar_t > & getline (basic_istream< wchar_t > &, wstring &)`
- `template basic_istream< wchar_t > & getline (basic_istream< wchar_t > &, wstring &, wchar_t)`
- `template basic_istream< char > & getline (basic_istream< char > &, string &)`
- `template basic_istream< char > & getline (basic_istream< char > &, string &, char)`
- `template<>`
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<>`
`basic_istream< char > & getline (basic_istream< char > &__in, basic_string< char > &__str, char __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<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 _Facet >`
`bool has_facet (const locale &__loc) throw ()`
- `template bool has_facet< __timepunct< char > > (const locale &)`
- `template bool has_facet< __timepunct< wchar_t > > (const locale &)`
- `template bool has_facet< codecvt< char, char, mbstate_t > > (const locale &)`
- `template bool has_facet< codecvt< wchar_t, char, mbstate_t > > (const locale &)`
- `template bool has_facet< collate< char > > (const locale &)`
- `template bool has_facet< collate< wchar_t > > (const locale &)`
- `template bool has_facet< ctype< char > > (const locale &)`
- `template bool has_facet< ctype< wchar_t > > (const locale &)`
- `template bool has_facet< messages< char > > (const locale &)`
- `template bool has_facet< messages< wchar_t > > (const locale &)`
- `template bool has_facet< money_get< char > > (const locale &)`
- `template bool has_facet< money_get< wchar_t > > (const locale &)`
- `template bool has_facet< money_put< char > > (const locale &)`

- template bool **has_facet**< **money_put**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **moneypunct**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **moneypunct**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **num_get**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **num_get**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **num_put**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **num_put**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **numpunct**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **numpunct**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **time_get**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **time_get**< **wchar_t** > > (const [locale](#) &)
- template bool **has_facet**< **time_put**< **char** > > (const [locale](#) &)
- template bool **has_facet**< **time_put**< **wchar_t** > > (const [locale](#) &)
- [ios_base](#) & [hex](#) ([ios_base](#) & __base)
- template<typename [_Tp](#) >
 [__gnu_cxx::__promote](#)< [_Tp](#) >::[__type imag](#) ([_Tp](#))
- template<typename [_Tp](#) >
 [_Tp imag](#) (const [complex](#)< [_Tp](#) > & [_z](#))
- template<typename [_InputIterator1](#) , typename [_InputIterator2](#) , typename [_Compare](#) >
 bool [includes](#) ([_InputIterator1](#) [__first1](#) , [_InputIterator1](#) [__last1](#) , [_InputIterator2](#) [__first2](#) , [_InputIterator2](#) [__last2](#) , [_Compare](#) [__comp](#))
- template<typename [_InputIterator1](#) , typename [_InputIterator2](#) >
 bool [includes](#) ([_InputIterator1](#) [__first1](#) , [_InputIterator1](#) [__last1](#) , [_InputIterator2](#) [__first2](#) , [_InputIterator2](#) [__last2](#))
- template<typename [_Iter1](#) , typename [_Iter2](#) , typename [_Compare](#) >
 bool [includes](#) ([_Iter1](#) , [_Iter1](#) , [_Iter2](#) , [_Iter2](#) , [_Compare](#))
- template<typename [_Iter1](#) , typename [_Iter2](#) >
 bool [includes](#) ([_Iter1](#) , [_Iter1](#) , [_Iter2](#) , [_Iter2](#))
- 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](#) , typename [_Compare](#) >
 void [inplace_merge](#) ([_BIter](#) , [_BIter](#) , [_BIter](#) , [_Compare](#))

- `template<typename _BIter >`
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > inserter (_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, typename _Compare >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _InputIterator, typename _Predicate >`
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _FIter, typename _Compare >`
`bool is_sorted (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`
`bool is_sorted (_FIter, _FIter)`
- `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 _Filter, typename _Compare >`
`_Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`
`_Filter is_sorted_until (_Filter, _Filter)`
- `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 _Filter1, typename _Filter2 >`
`void iter_swap (_Filter1, _Filter2)`
- `template<typename _Tp >`
`_Tp kill_dependency (_Tp __y)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ld-`
`exp (_Tp __x, int __exp)`
- `long double ldexp (long double __x, int __exp)`
- `float ldexp (float __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `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 _IIter1, typename _IIter2, typename _Compare >`
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)`
- `template<typename _IIter1, typename _IIter2 >`
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2)`
- `template<typename _L1, typename _L2, typename... _L3 >`
`void lock (_L1 &, _L2 &, _L3 &...)`
- `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)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`
- `long double log (long double __x)`
- `float log (float __x)`
- `template<typename _Tp >`
`complex< _Tp > log (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)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log10 (_Tp __x)`
- `long double log10 (long double __x)`
- `float log10 (float __x)`
- `template<typename _Tp >`
`complex< _Tp > log10 (const complex< _Tp > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &)`

- [error_code](#) **make_error_code** (errc __e)
- [error_code](#) **make_error_code** (future_errc __errc)
- [error_condition](#) **make_error_condition** (errc __e)
- [error_condition](#) **make_error_condition** (future_errc __errc)
- template<typename _RandomAccessIterator, typename _Compare >
void [make_heap](#) (_RandomAccessIterator __first, _RandomAccessIterator __-
last, _Compare __comp)
- template<typename _RandomAccessIterator >
void [make_heap](#) (_RandomAccessIterator __first, _RandomAccessIterator __-
last)
- template<typename _RAIter, typename _Compare >
void **make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter >
void **make_heap** (_RAIter, _RAIter)
- 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, typename _Compare >
_Tp **max** ([initializer_list](#)< _Tp >, _Compare)
- template<typename _Tp >
_Tp **max** ([initializer_list](#)< _Tp >)
- template<typename _Tp, typename _Compare >
const _Tp & **max** (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
const _Tp & **max** (const _Tp &__a, const _Tp &__b)
- template<typename _ForwardIterator, typename _Compare >
_ForwardIterator [max_element](#) (_ForwardIterator __first, _ForwardIterator __-
last, _Compare __comp)
- template<typename _ForwardIterator >
_ForwardIterator [max_element](#) (_ForwardIterator __first, _ForwardIterator __-
last)
- template<typename _FIter, typename _Compare >
_FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter >
_FIter **max_element** (_FIter, _FIter)
- template<typename _Tp, typename _Class >
_Mem_fn< _Tp _Class::* > [mem_fn](#) (_Tp _Class::* __pm)

- `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_t< _Ret, _Tp > mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::*__f)())`
- `void * memchr (void * __s, int __c, size_t __n)`
- `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 _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _Tp, typename _Compare >`
`_Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`_Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _FIter, typename _Compare >`
`_FIter min_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`
`_FIter min_element (_FIter, _FIter)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`

- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b,`
`_Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `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 _FIter, typename _Compare >`
`pair< _FIter, _FIter > minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`
`pair< _FIter, _FIter > minmax_element (_FIter, _FIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`
`pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `long double modf (long double __x, long double *__iptr)`
- `float modf (float __x, float *__iptr)`
- `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 _II, typename _OI >`
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && move (_Tp &&__t)`
- `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 >`
`_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 _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, 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, typename _Compare >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter, typename _Compare >`
`bool next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`
`bool next_permutation (_BIter, _BIter)`
- `ios_base & nboolalpha (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 >`
`__gnu_cxx::__promote<_Tp >::__type norm (_Tp __x)`
- `template<typename _Tp >`
`_Tp norm (const complex<_Tp > &)`
- `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 & nounitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`

- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `ios_base & oct (ios_base &__base)`
- `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<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<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 _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 _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 _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_-`
`ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const`
`istreambuf_iterator< _CharT, _Traits > &__b)`
- `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 _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__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 _Val >`
`bool operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _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 _Seq >`
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _T1 , class _T2 >`
`bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _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 _Tp , typename _Compare , typename _Alloc >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp , typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__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)`
- `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 _Ref, typename _Ptr >`
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _StateT >`
`bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__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 _Fwd_list_iterator< _Tp, _Alloc > &__x, const _Fwd_list_const_iterator< _Tp, _Alloc > &__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 _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 _T1, typename _T2 >`
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Bi_iter, class _Allocator >`
`bool operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `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 >`
`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 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 operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs)`
- `template<typename _Biter >`
`bool operator!= (const sub_match< _Biter > & __lhs, const sub_match< _Biter > & __rhs)`
- `template<typename _Tp, std::size_t _Nm >`
`bool operator!= (const array< _Tp, _Nm > & __one, const array< _Tp, _Nm > & __two)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const _Tp & __t, const valarray< _Tp > & __v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray< _Tp > & __v, const _Tp & __t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray< _Tp > & __v, const valarray< _Tp > & __w)`
- `template<typename... _TElements, typename... _UElements >`
`bool operator!= (const tuple< _TElements... > & __t, const tuple< _TElements... > & __u)`
- `bool operator!= (thread::id __x, thread::id __y)`
- `bool operator!= (const error_condition & __lhs, const error_condition & __rhs)`
- `bool operator!= (const error_condition & __lhs, const error_code & __rhs)`
- `bool operator!= (const error_code & __lhs, const error_condition & __rhs)`
- `bool operator!= (const error_code & __lhs, const error_code & __rhs)`
- `template<typename _Res, typename... _Args >`
`bool operator!= (_M_clear_type *, const function< _Res(_Args...) > & __f)`
- `template<typename _Res, typename... _Args >`
`bool operator!= (const function< _Res(_Args...) > & __f, _M_clear_type *)`

- `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 _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, _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<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 _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<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `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, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `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, _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, _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, _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<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >`, typename `__fun< __bitwise_and, typename _Dom1::value_type >::result_type >` **operator&** (const `_Expr< _Dom1, typename _Dom1::value_type > &_v`, const `_Expr< _Dom2, typename _Dom2::value_type > &_w`)
 - `_Ios_Iostate operator&` (`_Ios_Iostate __a, _Ios_Iostate __b`)
 - `_Ios_Openmode operator&` (`_Ios_Openmode __a, _Ios_Openmode __b`)
 - `_Ios_Fmtflags operator&` (`_Ios_Fmtflags __a, _Ios_Fmtflags __b`)
 - `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >`, typename `__fun< __bitwise_and, _Tp >::result_type >` **operator&** (const `_Tp &_t`, const `valarray< _Tp > &_v`)
 - `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >`, typename `__fun< __bitwise_and, _Tp >::result_type >` **operator&** (const `valarray< _Tp > &_v`, const `_Tp &_t`)
 - `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >`, typename `__fun< __bitwise_and, _Tp >::result_type >` **operator&** (const `valarray< _Tp > &_v`, const `valarray< _Tp > &_w`)
 - `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, _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, _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)`
- `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<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, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `_Ios_Iostate & operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Openmode & operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Fmtflags & operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `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 _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<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, _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 _Dom1, class _Dom2 >
_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __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<typename _Tp >
_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > **operator*** (const _Tp &__t, const [valarray](#)< _Tp > &__v)
 - 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, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > **operator*** (const [valarray](#)< _Tp > &__v, const [valarray](#)< _Tp > &__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, _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)
 - 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<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 _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<typename _Iterator >`
`move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator`
`>::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator`
`>::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const _Deque_-`
`iterator< _Tp, _Ref, _Ptr > &__x)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `_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, _CharT __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+ (_CharT __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 _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 _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __-`
`fun< __plus, _Tp >::result_type > operator+ (const _Tp &__t, const valarray<`
`_Tp > &__v)`
- `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 _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 _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `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<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 _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<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 _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<typename _IteratorL, typename _IteratorR >`
`auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator<`
`_IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type operator- (const reverse_`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `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<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)`

- `ptrdiff_t operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)
- `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 _Tp > _Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _fun< __minus, _Tp >::result_type > operator-` (`const valarray< _Tp > &__v`, `const _Tp &__t`)
- `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 > complex< _Tp > operator-` (`const complex< _Tp > &__x`)
- `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<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, _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 > _Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _fun< __divides, _Tp >::result_type > operator/` (`const _Tp &__t`, `const valarray< _Tp > &__v`)

- `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, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &_`
`_v, const valarray< _Tp > &_w)`
- `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<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<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<typename _Tp , typename _Tp_Deleter , typename _Up , typename _Up_Deleter >`
`bool operator< (const unique_ptr< _Tp, _Tp_Deleter > &_x, const unique_`
`ptr< _Up, _Up_Deleter > &_y)`
- `template<typename _Tp , typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &_x, const vector< _Tp, _Alloc`
`> &_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 _Tp, typename _Seq >`
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _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 _Seq >`
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<class _T1, class _T2 >`
`bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__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 _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 _Tp, typename _Compare, typename _Alloc >`
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_-`
`iterator< _IteratorR > &__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)`
- `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 _Ref, typename _Ptr >`
`bool operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _-`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`

- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `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`
`_CharT *__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 _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 >`
`bool 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 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 operator< (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _BiIter >`
`bool operator< (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm >`
`&__b)`
- `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 >`
`_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, _ValArray, _ValArray, _Tp, _Tp >, typename __-`
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator< (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs)`
- `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<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<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<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<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 _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __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<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 _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 piecewise_constant_distribution< _RealType >
&__x)`
 - `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 _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-
CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
 - `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 _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 _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-
CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`

 - `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 _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-
CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)`
 - `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 _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-
CharT, _Traits > &__os, const poisson_distribution< _IntType > &__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 _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 _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 _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 _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 _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](#) > &, const [std::extreme_value_distribution](#)< [_RealType](#) > &)
 - `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<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 _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](#) > &, const [std::uniform_real_distribution](#)< [_RealType](#) > &)
 - `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 _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 wostream & operator<< (wostream &, const wchar_t *)`
- `template wostream & operator<< (wostream &, wchar_t)`
- `template ostream & operator<< (ostream &, const signed char *)`
- `template ostream & operator<< (ostream &, const unsigned char *)`
- `template ostream & operator<< (ostream &, const char *)`
- `template ostream & operator<< (ostream &, signed char)`
- `template ostream & operator<< (ostream &, unsigned char)`
- `template ostream & operator<< (ostream &, char)`
- `template basic_ostream< wchar_t > & operator<< (basic_ostream< wchar_t > &, const wstring &)`
- `template basic_ostream< char > & operator<< (basic_ostream< char > &, const string &)`
- `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 _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<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 _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 _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<class _CharT, class _Traits > basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &_out, thread::id __id)`
- `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &_os, const error_code &_e)`
- `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, typename _MoneyT > basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &_os, _Put_money< _MoneyT > _f)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setw __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, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Resetiosflags __f)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _`
`Traits > &__os, const complex< _Tp > &__x)`
- `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<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 _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<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<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<typename `_Tp`, typename `_Tp_Deleter`, typename `_Up`, typename `_Up_Deleter` >
bool **operator**<= (const `unique_ptr`< `_Tp`, `_Tp_Deleter` > &__x, const `unique_ptr`< `_Up`, `_Up_Deleter` > &__y)
 - template<typename `_Tp`, typename `_Alloc` >
bool **operator**<= (const `vector`< `_Tp`, `_Alloc` > &__x, const `vector`< `_Tp`, `_Alloc` > &__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 `_Tp`, typename `_Seq` >
bool **operator**<= (const `stack`< `_Tp`, `_Seq` > &__x, const `stack`< `_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 `_Seq` >
bool **operator**<= (const `queue`< `_Tp`, `_Seq` > &__x, const `queue`< `_Tp`, `_Seq` > &__y)
 - template<class `_T1`, class `_T2` >
bool **operator**<= (const `pair`< `_T1`, `_T2` > &__x, const `pair`< `_T1`, `_T2` > &__y)
 - template<typename `_Key`, typename `_Compare`, typename `_Alloc` >
bool **operator**<= (const `multiset`< `_Key`, `_Compare`, `_Alloc` > &__x, const `multiset`< `_Key`, `_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 `_Tp`, typename `_Compare`, typename `_Alloc` >
bool **operator**<= (const `map`< `_Key`, `_Tp`, `_Compare`, `_Alloc` > &__x, const `map`< `_Key`, `_Tp`, `_Compare`, `_Alloc` > &__y)
 - template<typename `_Tp`, typename `_Alloc` >
bool **operator**<= (const `list`< `_Tp`, `_Alloc` > &__x, const `list`< `_Tp`, `_Alloc` > &__y)
 - template<typename `_IteratorL`, typename `_IteratorR` >
bool **operator**<= (const `move_iterator`< `_IteratorL` > &__x, const `move_iterator`< `_IteratorR` > &__y)
 - template<typename `_IteratorL`, typename `_IteratorR` >
bool **operator**<= (const `reverse_iterator`< `_IteratorL` > &__x, const `reverse_iterator`< `_IteratorR` > &__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)`
- `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 _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 forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `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 _CharT * __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 _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 >`
`bool 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 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 operator<= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Biter >`
`bool operator<= (const sub_match< _Biter > &__lhs, const sub_match< _Biter > &__rhs)`

- `template<typename _Tp, std::size_t _Nm>`
`bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `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 _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 >`
`_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<typename... _TElements, typename... _UElements>`
`bool operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `bool operator<= (thread::id __x, thread::id __y)`
- `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 _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 _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<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 _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 _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator==(const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool operator==(const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `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 _Alloc >`
`bool operator==(const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__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 _Val >`
`bool operator==(const Rb_tree_iterator< _Val > &__x, const Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator==(const stack< _Tp, _Seq > &__x, const stack< _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 _Seq >`
`bool operator==(const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _T1, class _T2 >`
`bool operator==(const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator==(const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _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 _Tp, typename _Compare, typename _Alloc >`
`bool operator==(const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator==(const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`

- `template<typename _Val >`
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__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)`
- `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 _Ref, typename _Ptr >`
`bool operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _StateT >`
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__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 _Fwd_list_iterator< _Tp, _Alloc > &__x, const _Fwd_list_const_iterator< _Tp, _Alloc > &__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 _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__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 _T1, typename _T2 >`
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool operator== (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `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 >`
`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 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 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 operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Tp, std::size_t _Nm >`
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `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 >`
`_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 >`
`_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... _TElements, typename... _UElements>`
`bool operator== (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs)`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)`
- `bool operator== (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _Res, typename... _Args>`
`bool operator== (_M_clear_type *, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>`
`bool operator== (const function< _Res(_Args...)> &__f, _M_clear_type *)`
- `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, _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 _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<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 _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 _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_-`
`ptr< _Up, _Up_Deleter > &__y)`

- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__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 _Tp, typename _Seq >`
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _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 _Seq >`
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _T1, class _T2 >`
`bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _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 _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__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)`
- `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 _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 [forward_list](#)< _Tp, _Alloc > &__lx, const [forward_list](#)< _Tp, _Alloc > &__ly)
 - 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 _CharT *__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 _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 >
bool **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 **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 **operator**> (const [basic_string](#)< typename [iterator_traits](#)< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const [sub_match](#)< _Bi_iter > &__rhs)
 - template<typename _BiIter >
bool **operator**> (const [sub_match](#)< _BiIter > &__lhs, const [sub_match](#)< _BiIter > &__rhs)
 - template<typename _Tp, std::size_t _Nm >
bool **operator**> (const [array](#)< _Tp, _Nm > &__one, const [array](#)< _Tp, _Nm > &__two)

- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`fun< __greater, _Tp >::result_type > operator> (const _Tp &__t, const valarray`
`< _Tp > &__v)`
- `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 _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... _TElements, typename... _UElements>`
`bool operator> (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `bool operator> (thread::id __x, thread::id __y)`
- `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<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<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<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<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<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__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 _Tp, typename _Seq >`
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _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 _Seq >`
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _T1, class _T2 >`
`bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _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 _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__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)`
- `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 _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 forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `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 _CharT * __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 _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 >`
`bool 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 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 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 operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`

- `template<typename _Tp, std::size_t _Nm>`
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > 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 > operator>= (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `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<typename... _TElements, typename... _UElements>`
`bool operator>= (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `bool operator>= (thread::id __x, thread::id __y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename,`
`typename > class _Base>`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`
`&__str)`
- `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<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<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<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<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 _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &_is, piecewise_linear_distribution< _RealType > &_x)`
- `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 _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &_is, discrete_distribution< _IntType > &_x)`
- `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 _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &_is, student_t_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 _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &_is, chi_squared_distribution< _RealType > &_x)`
- `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, normal_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 _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &_is, poisson_distribution< _IntType > &_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 _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 _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 _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 _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 _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<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `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 > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`

- template [wistream](#) & **operator**>> ([wistream](#) &, wchar_t *)
- template [wistream](#) & **operator**>> ([wistream](#) &, wchar_t &)
- template [istream](#) & **operator**>> ([istream](#) &, signed char *)
- template [istream](#) & **operator**>> ([istream](#) &, unsigned char *)
- template [istream](#) & **operator**>> ([istream](#) &, signed char &)
- template [istream](#) & **operator**>> ([istream](#) &, unsigned char &)
- template [istream](#) & **operator**>> ([istream](#) &, char *)
- template [istream](#) & **operator**>> ([istream](#) &, char &)
- template [basic_istream](#)< wchar_t > & **operator**>> ([basic_istream](#)< wchar_t > &, [wstring](#) &)
- template [basic_istream](#)< char > & **operator**>> ([basic_istream](#)< char > &, [string](#) &)
- template<>
[basic_istream](#)< char > & **operator**>> ([basic_istream](#)< char > &__is, [basic_string](#)< char > &__str)
- 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 _Tp >
_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > **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 > **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 > **operator**>> (const [valarray](#)< _Tp > &__v, const [valarray](#)< _Tp > &__w)
- template<typename _CharT, typename _Traits, typename _Tp >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &&__is, _Tp &__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 _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, _Setfill< _CharT > __f)

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `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<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<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<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 _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)`
- `_Ios_Iostate operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Openmode operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `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)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `_Ios_Iostate & operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Openmode & operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Fmtflags & operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `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<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)`
- `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, _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 _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)`
- `_Ios_Iostate operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Openmode operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_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<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)`
- `_Ios_Iostate & operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Openmode & operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Fmtflags & operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `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, _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)`
- `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<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, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > 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 > operator|| (const valarray< _Tp >`
`&__v, const _Tp &__t)`

- `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)`
- `_Ios_Iostate operator~ (_Ios_Iostate __a)`
- `_Ios_Openmode operator~ (_Ios_Openmode __a)`
- `_Ios_Fmtflags operator~ (_Ios_Fmtflags __a)`
- `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 _RAIter, typename _Compare >`
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `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 _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last)`
- `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 _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 _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred)`
- `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`
- `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 _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _FIter, typename _Predicate >`
`_FIter partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _Tp >`
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `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<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<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<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)`

- `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, 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)`
- `float pow (float _x, float _y)`
- `template<typename _Tp >`
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _BidirectionalIterator >`
`_BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_`
`traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator _-`
`last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator _-`
`last)`
- `template<typename _BIter, typename _Compare >`
`bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`
`bool prev_permutation (_BIter, _BIter)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp _x)`
- `template<typename _Tp >`
`std::complex< _Tp > proj (const std::complex< _Tp > &)`

- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__-`
`x)(_Arg1, _Arg2))`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __-`
`intl=false)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _RandomNumberGenerator &__rand)`
- `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 _Tp >`
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp >`
`_Tp real (const complex< _Tp > &__z)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _FIter, typename _Tp >`
`_FIter remove (_FIter, _FIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__value)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_IIter, _IIter, _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 _FIter, typename _Predicate >`
`_FIter remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `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 _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex &__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 _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _BIter, typename _OIter >`
`_OIter reverse_copy (_BIter, _BIter, _OIter)`

- [ios_base & right](#) ([ios_base](#) & __base)
- `template<typename _ForwardIterator >`
`void rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last)`
- `template<typename _Filter >`
`void rotate (_Filter, _Filter, _Filter)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __-`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _Filter, typename _OIter >`
`_OIter rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- [ios_base & scientific](#) ([ios_base](#) & __base)
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`
`__predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `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 _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 _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, typename _Compare >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `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 _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `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, type-`
`name _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `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 _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)`
- `long double sin (long double __x)`
- `float sin (float __x)`
- `template<typename _Tp >`
`complex< _Tp > sin (const complex< _Tp > &)`
- `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 >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`sinh (_Tp __x)`
- `long double sinh (long double __x)`
- `float sinh (float __x)`
- `template<typename _Tp >`
`complex< _Tp > sinh (const complex< _Tp > &)`
- `ios_base & skipws (ios_base &__base)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void sort (_RAIter, _RAIter)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_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 _Tp >`
`_Expr<_UnClos<_Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray<_Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr<_UnClos<_Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt`
`(const _Expr<_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer<_Tp >::__value, double >::__type`
`sqrt (_Tp __x)`
- `long double sqrt (long double __x)`
- `float sqrt (float __x)`
- `template<typename _Tp >`
`complex<_Tp > sqrt (const complex<_Tp > &)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __-`
`__last, _Predicate __pred)`
- `template<typename _BIter, typename _Predicate >`
`_BIter stable_partition (_BIter, _BIter, _Predicate)`
- `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, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr<_Tp > static_pointer_cast (const shared_ptr<_Tp1 > &__r)`
- `double stof (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<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<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<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<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<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<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<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<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 _Tp_Deleter >
void **swap** ([unique_ptr](#)< _Tp, _Tp_Deleter > &__x, [unique_ptr](#)< _Tp, _Tp_-Deleter > &__y)
- template<typename _Tp, typename _Alloc >
void **swap** ([vector](#)< _Tp, _Alloc > &__x, [vector](#)< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >

```

void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x,
_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
• template<typename _Tp, typename _Seq >
void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _-
Alloc > &__y)
• template<typename _Tp, typename _Sequence, typename _Compare >
void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_-
queue< _Tp, _Sequence, _Compare > &__y)
• template<typename _Tp, typename _Seq >
void swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)
• template<class _T1, class _T2 >
void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _-
Compare, _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 _Key, typename _Tp, typename _Compare, typename _Alloc >
void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp,
_Compare, _Alloc > &__y)
• template<typename _Tp, typename _Alloc >
void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)
• template<typename _Tp, typename _Alloc >
void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)
• template<typename _Tp >
void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)
• template<typename _Tp >
void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)
• template<typename _Tp, typename _Alloc >
void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc >
&__ly)
• template<typename _CharT, typename _Traits, typename _Alloc >
void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _-
CharT, _Traits, _Alloc > &__rhs)
• template<typename _Tp, size_t _Nm>
void swap (_Tp(&)[_Nm], _Tp(&)[_Nm])
• template<typename _Tp >
void swap (_Tp &__a, _Tp &__b)
• template<typename _Bi_iter, typename _Allocator >
void swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results< _-
Bi_iter, _Allocator > &__rhs)

```

- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `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)`
- `void swap (thread &__x, thread &__y)`
- `template<typename _Mutex >`
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<typename _Res, typename... _ArgTypes>`
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`
- `template<typename _Res >`
`void swap (promise< _Res > &__x, promise< _Res > &__y)`
- `template<typename _Res, typename... _Args>`
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `_GLIBCXX_CONST const error_category & system_category () throw ()`
- `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)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tan (_Tp __x)`
- `long double tan (long double __x)`
- `float tan (float __x)`
- `template<typename _Tp >`
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `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)`
- `long double tanh (long double __x)`
- `float tanh (float __x)`
- `template<typename _Tp >`
`complex< _Tp > tanh (const complex< _Tp > &)`
- `void terminate () __attribute__((__noreturn__)) throw ()`
- `size_t const string &__grouping_tmp throw ()`
- `template<typename _Ex >`
`void throw_with_nested (_Ex __ex)`
- `template<typename... _Elements>`
`tuple< _Elements &...> tie (_Elements &...__args)`
- `string to_string (long double __val)`
- `string to_string (double __val)`
- `string to_string (float __val)`
- `string to_string (unsigned long long __val)`
- `string to_string (long long __val)`
- `string to_string (unsigned long __val)`
- `string to_string (long __val)`
- `string to_string (unsigned __val)`
- `string to_string (int __val)`
- `wstring to_wstring (long double __val)`
- `wstring to_wstring (double __val)`
- `wstring to_wstring (float __val)`
- `wstring to_wstring (unsigned long long __val)`
- `wstring to_wstring (long long __val)`
- `wstring to_wstring (unsigned long __val)`
- `wstring to_wstring (long __val)`
- `wstring to_wstring (unsigned __val)`
- `wstring to_wstring (int __val)`
- `template<typename _CharT >`
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , type-`
`name _BinaryOperation >`
`_OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`
`op)`
- `template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >`
`_OutputIterator transform (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _UnaryOperation __unary_op)`

- `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 _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 _ForwardIterator, typename _BinaryPredicate > _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator > _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _FIter, typename _BinaryPredicate > _FIter unique (_FIter, _FIter, _BinaryPredicate)`
- `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, typename _Compare >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &)`
- `ios_base & uppercase (ios_base &__base)`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &__loc)`
- `template const __timepunct< char > & use_facet< __timepunct< char > >`
`(const locale &)`
- `template const __timepunct< wchar_t > & use_facet< __timepunct< wchar_t`
`> > (const locale &)`
- `template const codecvt< char, char, mbstate_t > & use_facet< codecvt< char,`
`char, mbstate_t > > (const locale &)`
- `template const codecvt< wchar_t, char, mbstate_t > & use_facet< codecvt<`
`wchar_t, char, mbstate_t > > (const locale &)`
- `template const collate< char > & use_facet< collate< char > > (const locale`
`&)`
- `template const collate< wchar_t > & use_facet< collate< wchar_t > > (const`
`locale &)`
- `template const ctype< char > & use_facet< ctype< char > > (const locale &)`
- `template const ctype< wchar_t > & use_facet< ctype< wchar_t > > (const`
`locale &)`
- `template const messages< char > & use_facet< messages< char > > (const`
`locale &)`
- `template const messages< wchar_t > & use_facet< messages< wchar_t > >`
`(const locale &)`
- `template const money_get< char > & use_facet< money_get< char > >`
`(const locale &)`

- template const [money_get](#)< wchar_t > & [use_facet](#)< [money_get](#)< wchar_t > > (const [locale](#) &)
- template const [money_put](#)< char > & [use_facet](#)< [money_put](#)< char > > (const [locale](#) &)
- template const [money_put](#)< wchar_t > & [use_facet](#)< [money_put](#)< wchar_t > > (const [locale](#) &)
- template const [moneypunct](#)< char, false > & [use_facet](#)< [moneypunct](#)< char, false > > (const [locale](#) &)
- template const [moneypunct](#)< char, true > & [use_facet](#)< [moneypunct](#)< char, true > > (const [locale](#) &)
- template const [moneypunct](#)< wchar_t, false > & [use_facet](#)< [moneypunct](#)< wchar_t, false > > (const [locale](#) &)
- template const [moneypunct](#)< wchar_t, true > & [use_facet](#)< [moneypunct](#)< wchar_t, true > > (const [locale](#) &)
- template const [num_get](#)< char > & [use_facet](#)< [num_get](#)< char > > (const [locale](#) &)
- template const [num_get](#)< wchar_t > & [use_facet](#)< [num_get](#)< wchar_t > > (const [locale](#) &)
- template const [num_put](#)< char > & [use_facet](#)< [num_put](#)< char > > (const [locale](#) &)
- template const [num_put](#)< wchar_t > & [use_facet](#)< [num_put](#)< wchar_t > > (const [locale](#) &)
- template const [numpunct](#)< char > & [use_facet](#)< [numpunct](#)< char > > (const [locale](#) &)
- template const [numpunct](#)< wchar_t > & [use_facet](#)< [numpunct](#)< wchar_t > > (const [locale](#) &)
- template const [time_get](#)< char > & [use_facet](#)< [time_get](#)< char > > (const [locale](#) &)
- template const [time_get](#)< wchar_t > & [use_facet](#)< [time_get](#)< wchar_t > > (const [locale](#) &)
- template const [time_put](#)< char > & [use_facet](#)< [time_put](#)< char > > (const [locale](#) &)
- template const [time_put](#)< wchar_t > & [use_facet](#)< [time_put](#)< wchar_t > > (const [locale](#) &)
- wchar_t * [wcschr](#) (wchar_t * __p, wchar_t __c)
- wchar_t * [wspbrk](#) (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 [istream](#) & [ws](#) ([istream](#) &)
- template<typename _CharT, typename _Traits > [basic_istream](#)< _CharT, _Traits > & [ws](#) ([basic_istream](#)< _CharT, _Traits > & __is)

- `template<typename _Tp >`
[reference_wrapper](#)< const _Tp > [cref](#) ([reference_wrapper](#)< _Tp > __t)
- `template<typename _Tp >`
[reference_wrapper](#)< const _Tp > [cref](#) (const _Tp &__t)
- `template<typename _Tp >`
 bool [operator!=](#) (const _Tp &__x, const [complex](#)< _Tp > &__y)
- `template<typename _Tp >`
 bool [operator!=](#) (const [complex](#)< _Tp > &__x, const _Tp &__y)
- `template<typename _Tp >`
 bool [operator!=](#) (const [complex](#)< _Tp > &__x, const [complex](#)< _Tp > &__y)
- `template<size_t _Nb >`
[bitset](#)< _Nb > [operator&](#) (const [bitset](#)< _Nb > &__x, const [bitset](#)< _Nb > &__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 _Tp &__y)
- `template<typename _Tp >`
[complex](#)< _Tp > [operator*](#) (const [complex](#)< _Tp > &__x, const [complex](#)< _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 _Tp &__y)
- `template<typename _Tp >`
[complex](#)< _Tp > [operator+](#) (const [complex](#)< _Tp > &__x, const [complex](#)< _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 _Tp &__y)
- `template<typename _Tp >`
[complex](#)< _Tp > [operator-](#) (const [complex](#)< _Tp > &__x, const [complex](#)< _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 _Tp &__y)
- `template<typename _Tp >`
[complex](#)< _Tp > [operator/](#) (const [complex](#)< _Tp > &__x, const [complex](#)< _Tp > &__y)
- `template<class _Traits >`
[basic_ostream](#)< char, _Traits > & [operator<<](#) ([basic_ostream](#)< char, _Traits > &__out, const unsigned 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 char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, unsigned 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, char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `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 >`
`bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>>> (basic_istream< char, _Traits > &__in, signed char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<>`
`basic_istream< char > & operator>>> (basic_istream< char > &__in, char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`

- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _CharT, class _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 >`
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (_Tp &__t)`

Matching, Searching, and Replacing

- `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 _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 _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, 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 _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 _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`>`  
`bool regex_match (_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 _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)`
  - `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 _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 _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_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_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 _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 _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)`
-

## Variables

- `size_t __grouping_size`
- `static ios_base::Init __ioint`
- `function< void()> __once_functor`
- `const adopt_lock_t adopt_lock`
- `const defer_lock_t defer_lock`
- `const error_category *const future_category`
- `error_code make_error_code (errc)`
- `error_condition make_error_condition (errc)`
- `const nothrow_t nothrow`
- `const try_to_lock_t try_to_lock`

### 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.

- `ostream cerr`
- `istream cin`
- `ostream clog`
- `ostream cout`
- `wostream wcerr`
- `wistream wcin`
- `wostream wclog`
- `wostream wcout`

### 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 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 88 of file `postypes.h`.

#### 4.11.2.3 typedef fpos<mbstate\_t> std::streampos

File position for char streams.

Definition at line 228 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 98 of file `postypes.h`.

#### 4.11.2.5 typedef fpos<mbstate\_t> std::u16streampos

File position for `char16_t` streams.

Definition at line 234 of file `postypes.h`.

#### 4.11.2.6 typedef fpos<mbstate\_t> std::u32streampos

File position for `char32_t` streams.

Definition at line 236 of file `postypes.h`.

#### 4.11.2.7 typedef fpos<mbstate\_t> std::wstreampos

File position for `wchar_t` streams.

Definition at line 230 of file `postypes.h`.

### 4.11.3 Enumeration Type Documentation

#### 4.11.3.1 anonymous enum

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more. This controls some aspect of the sort routines.

Definition at line 2163 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 170 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\_indeterminate* Self-explanatory.

*round\_toward\_zero* Self-explanatory.

*round\_to\_nearest* To the nearest representable value.

*round\_toward\_infinity* Self-explanatory.

*round\_toward\_neg\_infinity* Self-explanatory.

Definition at line 155 of file limits.



#### 4.11.4 Function Documentation

**4.11.4.1** `template<typename _RandomAccessIterator , typename _Compare  
> void std::__final_insertion_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function for the sort routine.

Definition at line 2183 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

**4.11.4.2** `template<typename _RandomAccessIterator > void  
std::__final_insertion_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last) [inline]`

This is a helper function for the sort routine.

Definition at line 2168 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

**4.11.4.3** `template<typename _RandomAccessIterator , typename _Tp >  
_RandomAccessIterator std::__find (_RandomAccessIterator  
__first, _RandomAccessIterator __last, const _Tp & __val,  
random_access_iterator_tag) [inline]`

This is an overload used by `find()` for the RAI case.

Definition at line 146 of file `stl_algo.h`.

**4.11.4.4** `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 124 of file `stl_algo.h`.

Referenced by `find()`.

**4.11.4.5** `template<typename _RandomAccessIterator , typename _Predicate  
> _RandomAccessIterator std::__find_if (_RandomAccessIterator  
__first, _RandomAccessIterator __last, _Predicate __pred,  
random_access_iterator_tag) [inline]`

This is an overload used by `find_if()` for the RAI case.

Definition at line 194 of file `stl_algo.h`.

**4.11.4.6** `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 135 of file `stl_algo.h`.

Referenced by `find_if()`.

**4.11.4.7** `template<typename _RandomAccessIterator , typename _Predicate >  
_RandomAccessIterator std::__find_if_not (_RandomAccessIterator  
__first, _RandomAccessIterator __last, _Predicate __pred,  
random_access_iterator_tag) [inline]`

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 254 of file `stl_algo.h`.

**4.11.4.8** `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 243 of file `stl_algo.h`.

Referenced by `find_if_not()`.

**4.11.4.9** `template<typename _EuclideanRingElement >  
_EuclideanRingElement std::__gcd (_EuclideanRingElement __m,  
_EuclideanRingElement __n) [inline]`

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 1483 of file `stl_algo.h`.

**4.11.4.10** `template<typename _RandomAccessIterator, typename _Compare  
> void std::__heap_select (_RandomAccessIterator __first,  
_RandomAccessIterator __middle, _RandomAccessIterator __last,  
_Compare __comp) [inline]`

This is a helper function for the sort routines.

Definition at line 1904 of file `stl_algo.h`.

References `make_heap()`.

**4.11.4.11** `template<typename _RandomAccessIterator > void  
std::__heap_select (_RandomAccessIterator __first,  
_RandomAccessIterator __middle, _RandomAccessIterator __last)  
[inline]`

This is a helper function for the sort routines.

Definition at line 1891 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

**4.11.4.12** `template<typename _ForwardIterator, typename  
_Predicate, typename _Distance > _ForwardIterator  
std::__inplace_stable_partition (_ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, _Distance __len)  
[inline]`

This is a helper function...

Definition at line 1768 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

**4.11.4.13** `template<typename _RandomAccessIterator, typename _Compare  
> void std::__inplace_stable_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function for the stable sorting routines.

Definition at line 3436 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.

**4.11.4.14** `template<typename _RandomAccessIterator > void  
std::__inplace_stable_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last) [inline]`

This is a helper function for the stable sorting routines.

Definition at line 3417 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

**4.11.4.15** `template<typename _RandomAccessIterator , typename _Compare  
> void std::__insertion_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function for the sort routine.

Definition at line 2114 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

**4.11.4.16** `template<typename _RandomAccessIterator > void  
std::__insertion_sort (_RandomAccessIterator __first,  
_RandomAccessIterator __last) [inline]`

This is a helper function for the sort routine.

Definition at line 2091 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

**4.11.4.17** `template<typename _RandomAccessIterator , typename  
_Size , typename _Compare > void std::__introsort_loop  
(_RandomAccessIterator __first, _RandomAccessIterator __last,  
_Size __depth_limit, _Compare __comp) [inline]`

This is a helper function for the sort routine.

Definition at line 2285 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

**4.11.4.18** `template<typename _RandomAccessIterator, typename _Size  
> void std::__introsort_loop (_RandomAccessIterator __first,  
_RandomAccessIterator __last, _Size __depth_limit) [inline]`

This is a helper function for the sort routine.

Definition at line 2263 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

**4.11.4.19** `template<typename _Functor, typename... _Args>  
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) [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`.

**4.11.4.20** `template<typename _Size > _Size std::__lg (_Size __n) [inline]`

This is a helper function for the sort routines. Precondition: `__n > 0`.

Definition at line 2307 of file `stl_algo.h`.

Referenced by `nth_element()`, `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`, and `sort()`.

**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) [inline]`

This is a helper function for the merge routines.

Definition at line 2955 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 _BidirectionalIterator , typename _Distance , typename _Pointer > void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size) [inline]`

This is a helper function for the merge routines.

Definition at line 2893 of file `stl_algo.h`.

References `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

**4.11.4.23** `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) [inline]`

This is a helper function for the merge routines.

Definition at line 2825 of file `stl_algo.h`.

**4.11.4.24** `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 > _BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result) [inline]`

This is a helper function for the merge routines.

Definition at line 2790 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

**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) [inline]`

This is a helper function for the merge routines.

Definition at line 3062 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 _BidirectionalIterator , typename _Distance >
void std::__merge_without_buffer (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2) [inline]
```

This is a helper function for the merge routines.

Definition at line 3018 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()`.

```
4.11.4.27 template<typename _Iterator , typename _Compare > void
std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator
__c, _Compare __comp) [inline]
```

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__a`.

Definition at line 96 of file stl\_algo.h.

References `iter_swap()`.

```
4.11.4.28 template<typename _Iterator > void std::__move_median_first
(_Iterator __a, _Iterator __b, _Iterator __c) [inline]
```

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 72 of file stl\_algo.h.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

```
4.11.4.29 template<typename _BidirectionalIterator , typename _Predicate
> _BidirectionalIterator std::__partition (_BidirectionalIterator
__first, _BidirectionalIterator __last, _Predicate __pred,
bidirectional_iterator_tag) [inline]
```

This is a helper function...

Definition at line 1738 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.30** `template<typename _ForwardIterator , typename _Predicate  
> _ForwardIterator std::__partition (_ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, forward_iterator_tag)  
[inline]`

This is a helper function...

Definition at line 1713 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

**4.11.4.31** `template<typename _RandomAccessIterator > void std::__reverse  
(_RandomAccessIterator __first, _RandomAccessIterator __last,  
random_access_iterator_tag) [inline]`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1403 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.32** `template<typename _BidirectionalIterator > void std::__reverse  
(_BidirectionalIterator __first, _BidirectionalIterator __last,  
bidirectional_iterator_tag) [inline]`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1383 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, and `reverse()`.

**4.11.4.33** `template<typename _RandomAccessIterator > void std::__rotate  
(_RandomAccessIterator __first, _RandomAccessIterator __middle,  
_RandomAccessIterator __last, random_access_iterator_tag)  
[inline]`

This is a helper function for the rotate algorithm.



Definition at line 1563 of file `stl_algo.h`.

References `iter_swap()`, `swap()`, and `swap_ranges()`.

```
4.11.4.34 template<typename _BidirectionalIterator > void std::__rotate
(_BidirectionalIterator __first, _BidirectionalIterator __middle,
_BidirectionalIterator __last, bidirectional_iterator_tag)
[inline]
```

This is a helper function for the rotate algorithm.

Definition at line 1533 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

```
4.11.4.35 template<typename _ForwardIterator > void std::__rotate
(_ForwardIterator __first, _ForwardIterator __middle,
_ForwardIterator __last, forward_iterator_tag) [inline]
```

This is a helper function for the rotate algorithm.

Definition at line 1497 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::M_deallocate_single_`-  
`object()`, and `rotate()`.

```
4.11.4.36 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)
[inline]
```

This is a helper function for the merge routines.

Definition at line 2861 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

**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) [inline]`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 445 of file `stl_algo.h`.

**4.11.4.38** `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) [inline]`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 406 of file `stl_algo.h`.

**4.11.4.39** `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)  
[inline]`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 352 of file `stl_algo.h`.

**4.11.4.40** `template<typename _ForwardIterator , typename _Integer  
, typename _Tp > _ForwardIterator std::__search_n  
(_ForwardIterator __first, _ForwardIterator __last, _Integer  
__count, const _Tp & __val, std::forward_iterator_tag) [inline]`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 320 of file `stl_algo.h`.

Referenced by `search_n()`.

**4.11.4.41** `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) [inline]`

This is a helper function...

Definition at line 1793 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

**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 2149 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

**4.11.4.43** `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 2136 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

**4.11.4.44** `template<typename _RandomAccessIterator, typename _Compare  
> void std::__unguarded_linear_insert (_RandomAccessIterator  
__last, _Compare __comp) [inline]`

This is a helper function for the sort routine.

Definition at line 2072 of file `stl_algo.h`.

**4.11.4.45** `template<typename _RandomAccessIterator > void  
std::__unguarded_linear_insert (_RandomAccessIterator __last)  
[inline]`

This is a helper function for the sort routine.

Definition at line 2054 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

**4.11.4.46** `template<typename _RandomAccessIterator , typename  
_Tp , typename _Compare > _RandomAccessIterator  
std::__unguarded_partition (_RandomAccessIterator __first,  
_RandomAccessIterator __last, const _Tp & __pivot, _Compare  
__comp) [inline]`

This is a helper function...

Definition at line 2219 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.47** `template<typename _RandomAccessIterator , typename  
_Tp > _RandomAccessIterator std::__unguarded_partition  
(_RandomAccessIterator __first, _RandomAccessIterator __last,  
const _Tp & __pivot) [inline]`

This is a helper function...

Definition at line 2199 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

**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 2252 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

```
4.11.4.49 template<typename _RandomAccessIterator >
 _RandomAccessIterator std::__unguarded_partition_pivot
 (_RandomAccessIterator __first, _RandomAccessIterator __last)
 [inline]
```

This is a helper function...

Definition at line 2240 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

```
4.11.4.50 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) [inline]
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward [iterator](#) as result.

Definition at line 1360 of file `stl_algo.h`.

```
4.11.4.51 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) [inline]
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output [iterator](#) as result.

Definition at line 1331 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) [inline]
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output [iterator](#) as result.

Definition at line 1302 of file `stl_algo.h`.

**4.11.4.53** `template<typename _InputIterator , typename _ForwardIterator  
> _ForwardIterator std::__unique_copy (_InputIterator __first,  
_InputIterator __last, _ForwardIterator __result, input_iterator_tag,  
forward_iterator_tag) [inline]`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward [iterator](#) as result.

Definition at line 1281 of file `stl_algo.h`.

**4.11.4.54** `template<typename _InputIterator , typename _OutputIterator  
> _OutputIterator std::__unique_copy (_InputIterator __first,  
_InputIterator __last, _OutputIterator __result, input_iterator_tag,  
output_iterator_tag) [inline]`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output [iterator](#) as result.

Definition at line 1258 of file `stl_algo.h`.

**4.11.4.55** `template<typename _ForwardIterator , typename _OutputIterator  
> _OutputIterator std::__unique_copy (_ForwardIterator  
__first, _ForwardIterator __last, _OutputIterator __result,  
forward_iterator_tag, output_iterator_tag) [inline]`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output [iterator](#) as result.

Definition at line 1235 of file `stl_algo.h`.

Referenced by `unique_copy()`.

**4.11.4.56** `template<typename _T1 , typename _T2 > void std::_Construct (_T1  
* __p, _T2 && __value) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 73 of file `stl_construct.h`.

**4.11.4.57** `template<typename _ForwardIterator > void std::_Destroy  
(_ForwardIterator __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 118 of file `stl_construct.h`.

**4.11.4.58** `template<typename _Tp > void std::_Destroy (_Tp * __pointer)`  
`[inline]`

Destroy the object pointed to by a pointer type.

Definition at line 88 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< result_type >::~~vector()`.

**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 142 of file `stl_numeric.h`.

**4.11.4.60** `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 116 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

**4.11.4.61** `template<typename _InputIterator , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
std::adjacent_difference (_InputIterator __first, _InputIterator  
__last, _OutputIterator __result, _BinaryOperation __binary_op)  
[inline]`

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 354 of file `stl_numeric.h`.

**4.11.4.62** `template<typename _InputIterator , typename _OutputIterator >  
_OutputIterator std::adjacent_difference (_InputIterator __first,  
_InputIterator __last, _OutputIterator __result) [inline]`

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 312 of file `stl_numeric.h`.

**4.11.4.63** `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 168 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()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

**4.11.4.64** `template<typename _Result , typename _Functor ,  
typename... _ArgTypes> _Bind_result<_Result, typename  
_Maybe_wrap_member_pointer<_Functor>::type (_ArgTypes...)>  
std::bind (_Functor __f, _ArgTypes... __args) [inline]`

Function template for [std::bind](#).

Definition at line 1390 of file functional.

#### **4.11.4.65 ios\_base& std::boolalpha (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::boolalpha).

Definition at line 794 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by noboolalpha().

#### **4.11.4.66 template<typename \_Tp > reference\_wrapper<const \_Tp> std::cref (reference\_wrapper<\_Tp > \_\_t) [inline]**

Partial specialization.

Definition at line 457 of file functional.

References cref().

#### **4.11.4.67 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 445 of file functional.

Referenced by cref().

#### **4.11.4.68 ios\_base& std::dec (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::dec, ios\_base::basefield).

Definition at line 932 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by operator>>().

#### **4.11.4.69 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 110 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()`, `std::sub_match<_Bi_iter >::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `partition_point()`, `std::match_results<_Bi_iter >::position()`, `__gnu_cxx::random_sample_n()`, and `upper_bound()`.

**4.11.4.70** `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 newline 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 541 of file `ostream`.

References `flush()`, `std::basic_ostream<_CharT, _Traits >::put()`, and `std::basic_ios<_CharT, _Traits >::widen()`.

**4.11.4.71** `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 552 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits >::put()`.

#### 4.11.4.72 `ios_base& std::fixed (ios_base & __base) [inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 957 of file `ios_base.h`.

References `std::ios_base::setf()`.

#### 4.11.4.73 `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 562 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits >::flush()`.

Referenced by `endl()`.

#### 4.11.4.74 `template<typename _Tp > enable_if<!is_lvalue_reference<_Tp>::value, _Tp&&>::type std::forward (typename std::identity<_Tp >::type && __t) [inline]`

Forward rvalues as rvalues.

Definition at line 59 of file `move.h`.

#### 4.11.4.75 `template<typename _Tp > enable_if<!is_lvalue_reference<_Tp>::value, _Tp&&>::type std::forward (typename std::identity<_Tp >::type & __t) [inline]`

forward (as per N2835) Forward lvalues as rvalues.

Definition at line 53 of file `move.h`.

#### 4.11.4.76 `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 256 of file `iomanip`.

**4.11.4.77** `template<typename _Tp > pair<_Tp*, ptrdiff_t>  
std::get_temporary_buffer (ptrdiff_t __len) [inline]`

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 85 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

**4.11.4.78** `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 2448 of file `vstring.h`.

References `std::basic_ios<_CharT, _Traits >::widen()`.

**4.11.4.79** `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) [inline]`

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 622 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()`.

```
4.11.4.80 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 2640 of file *basic\_string.h*.

References `std::basic_ios<_CharT, _Traits >::widen()`.

```
4.11.4.81 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) [inline]
```

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 1068 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()`.

#### 4.11.4.82 `template<typename _Facet > bool std::has_facet (const locale & __loc) throw () [inline]`

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 91 of file `locale_classes.tcc`.

#### 4.11.4.83 `ios_base& std::hex (ios_base & __base) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 940 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `std::regex_traits<_Ch_type >::value()`.

#### 4.11.4.84 `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 202 of file `stl_numeric.h`.

```
4.11.4.85 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 170 of file `stl_numeric.h`.

```
4.11.4.86 ios_base& std::internal (ios_base & __base) [inline]
```

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 907 of file `ios_base.h`.

References `std::ios_base::setf()`.

**4.11.4.87** `template<typename _ForwardIterator, typename _Tp> void  
std::iota(_ForwardIterator __first, _ForwardIterator __last, _Tp  
__value) [inline]`

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 81 of file `stl_numeric.h`.

**4.11.4.88** `template<typename _CharT> bool std::isalnum(_CharT __c, const  
locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

**4.11.4.89** `template<typename _CharT> bool std::isalpha(_CharT __c, const  
locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

**4.11.4.90** `template<typename _CharT> bool std::isctrl(_CharT __c, const  
locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

**4.11.4.91** `template<typename _CharT> bool std::isdigit(_CharT __c, const  
locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

**4.11.4.92** `template<typename _CharT> bool std::isgraph(_CharT __c, const  
locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

---

**4.11.4.93** `template<typename _CharT > bool std::islower (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

**4.11.4.94** `template<typename _CharT > bool std::isprint (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

**4.11.4.95** `template<typename _CharT > bool std::ispunct (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

**4.11.4.96** `template<typename _CharT > bool std::isspace (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

**4.11.4.97** `template<typename _CharT > bool std::isupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

**4.11.4.98** `template<typename _CharT > bool std::isxdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

**4.11.4.99** `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 915 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by operator `<<()`.

**4.11.4.100 ios\_base& std::noboolalpha (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::boolalpha).

Definition at line 802 of file ios\_base.h.

References boolalpha(), and std::ios\_base::unsetf().

**4.11.4.101 ios\_base& std::noshowbase (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::showbase).

Definition at line 818 of file ios\_base.h.

References showbase(), and std::ios\_base::unsetf().

**4.11.4.102 ios\_base& std::noshowpoint (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::showpoint).

Definition at line 834 of file ios\_base.h.

References showpoint(), and std::ios\_base::unsetf().

**4.11.4.103 ios\_base& std::noshowpos (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::showpos).

Definition at line 850 of file ios\_base.h.

References showpos(), and std::ios\_base::unsetf().

**4.11.4.104 ios\_base& std::noskipws (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::skipws).

Definition at line 866 of file ios\_base.h.

References skipws(), and std::ios\_base::unsetf().

**4.11.4.105 ios\_base& std::nounitbuf (ios\_base & *\_\_base*) [inline]**

Calls base.unsetf(ios\_base::unitbuf).

Definition at line 898 of file ios\_base.h.

References unitbuf(), and std::ios\_base::unsetf().

**4.11.4.106 ios\_base& std::nouppercase (ios\_base & \_\_base) [inline]**

Calls base.unsetf(ios\_base::uppercase).

Definition at line 882 of file ios\_base.h.

References std::ios\_base::unsetf(), and uppercase().

**4.11.4.107 ios\_base& std::oct (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::oct, ios\_base::basefield).

Definition at line 948 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by std::regex\_traits<\_Ch\_type >::value().

**4.11.4.108 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 1196 of file stl\_vector.h.

**4.11.4.109 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 259 of file stl\_stack.h.

**4.11.4.110 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 700 of file stl\_set.h.

**4.11.4.111** `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 294 of file `stl_queue.h`.

**4.11.4.112** `template<class _T1, class _T2 > bool std::operator!=(const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator==` to find the result.

Definition at line 171 of file `stl_pair.h`.

**4.11.4.113** `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 687 of file `stl_multiset.h`.

**4.11.4.114** `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 792 of file `stl_multimap.h`.

**4.11.4.115** `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 861 of file `stl_map.h`.

**4.11.4.116** `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 1519 of file stl\_list.h.

**4.11.4.117** `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 1853 of file stl\_deque.h.

**4.11.4.118** `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 1247 of file forward\_list.h.

**4.11.4.119** `template<typename _Tp, typename _Alloc > bool std::operator!=  
(const _Fwd_list_iterator< _Tp, _Alloc > & __x, const  
_Fwd_list_const_iterator< _Tp, _Alloc > & __y) [inline]`

Forward [list iterator](#) inequality comparison.

Definition at line 246 of file forward\_list.h.

**4.11.4.120** `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 2403 of file basic\_string.h.

**4.11.4.121** `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 2391 of file basic\_string.h.

**4.11.4.122** `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 2379 of file basic\_string.h.

**4.11.4.123** `template<typename _Res , typename... _Args> bool std::operator!=(  
_M_clear_type *, 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 2194 of file functional.



**4.11.4.124** `template<typename _Res, typename... _Args> bool std::operator!=(const function< _Res(_Args...)> & _f, _M_clear_type *)`  
`[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 2188 of file functional.

**4.11.4.125** `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 1366 of file bitset.

**4.11.4.126** `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 2315 of file basic\_string.h.

**4.11.4.127** `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 2299 of file basic\_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc >::append()`.

**4.11.4.128** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc > std::operator+ (_CharT  
__lhs, const basic_string<_CharT, _Traits, _Alloc > & __rhs)  
[inline]`

Concatenate character and string.

**Parameters:**

*lhs* First string.

*rhs* Last string.

**Returns:**

New string with *lhs* followed by *rhs*.

Definition at line 708 of file basic\_string.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**4.11.4.129** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc > std::operator+ (const  
_CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc > &  
__rhs) [inline]`

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 692 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**4.11.4.130** `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) [inline]`

Concatenate two strings.

**Parameters:**

*lhs* First string.

*rhs* Last string.

**Returns:**

New string with value of *lhs* followed by *rhs*.

Definition at line 2262 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::append()`.

**4.11.4.131** `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 1189 of file `stl_vector.h`.

References `lexicographical_compare()`.

```
4.11.4.132 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 253 of file `stl_stack.h`.

```
4.11.4.133 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 693 of file `stl_set.h`.

```
4.11.4.134 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 288 of file `stl_queue.h`.

```
4.11.4.135 template<class _T1 , class _T2 > 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 164 of file `stl_pair.h`.

```
4.11.4.136 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 680 of file `stl_multiset.h`.

**4.11.4.137** `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 785 of file `stl_multimap.h`.

**4.11.4.138** `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 854 of file `stl_map.h`.

**4.11.4.139** `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 1512 of file `stl_list.h`.

References `lexicographical_compare()`.

**4.11.4.140** `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 1845 of file `stl_deque.h`.

References `lexicographical_compare()`.

```
4.11.4.141 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 1239 of file `forward_list.h`.

References `lexicographical_compare()`.

```
4.11.4.142 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 2440 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.



**4.11.4.143** `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 2428 of file basic\_string.h.

**4.11.4.144** `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 2416 of file basic\_string.h.

**4.11.4.145** `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 2402 of file `vstring.h`.

**4.11.4.146** `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 2599 of file `basic_string.h`.

**4.11.4.147** `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 579 of file `ostream`.

**4.11.4.148** `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 526 of file ostream.

**4.11.4.149** `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 521 of file ostream.

**4.11.4.150** `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 508 of file ostream.

**4.11.4.151** `template<typename _CharT , typename _Traits > basic_ostream<  
_CharT, _Traits > & std::operator<< (basic_ostream< _CharT,  
_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 321 of file `ostream.tcc`.

References `std::ios_base::badbit`.

```
4.11.4.152 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 491 of file `ostream`.

```
4.11.4.153 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< (basic_ostream< char, _Traits > & __out,
unsigned char __c) [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 471 of file `ostream`.

```
4.11.4.154 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< (basic_ostream< char, _Traits > & __out, signed
char __c) [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 466 of file `ostream`.

```
4.11.4.155 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< (basic_ostream< char, _Traits > & __out, char
__c) [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 460 of file `ostream`.

```
4.11.4.156 template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<<
(basic_ostream<_CharT, _Traits > & __out, char __c) [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 454 of file `ostream`.

```
4.11.4.157 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 [22.2.2.2.2]). `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 449 of file `ostream`.

**4.11.4.158** `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) [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 1471 of file `bitset`.

References `std::__ctype_abstract_base< _CharT >::widen()`.

**4.11.4.159** `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 1208 of file `stl_vector.h`.



**4.11.4.160** `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`.

**4.11.4.161** `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 714 of file `stl_set.h`.

**4.11.4.162** `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 306 of file `stl_queue.h`.

**4.11.4.163** `template<class _T1, class _T2 > bool std::operator<= (const pair<`  
`_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 183 of file `stl_pair.h`.

**4.11.4.164** `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 701 of file `stl_multiset.h`.

**4.11.4.165** `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 806 of file stl\_multimap.h.

**4.11.4.166** `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 875 of file stl\_map.h.

**4.11.4.167** `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 1531 of file stl\_list.h.

**4.11.4.168** `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 1867 of file stl\_deque.h.

**4.11.4.169** `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.

---

**4.11.4.170** `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 2514 of file basic\_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`.

**4.11.4.171** `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 2502 of file basic\_string.h.

**4.11.4.172** `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 2490 of file basic\_string.h.

**4.11.4.173** `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 1172 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc >::begin()`, `std::vector<_Tp, _Alloc >::end()`, `equal()`, and `std::vector<_Tp, _Alloc >::size()`.

**4.11.4.174** `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 235 of file `stl_stack.h`.

```
4.11.4.175 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 676 of file `stl_set.h`.

```
4.11.4.176 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 270 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**4.11.4.177** `template<class _T1 , class _T2 > 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 158 of file `stl_pair.h`.

References `std::pair<_T1, _T2 >::first`, and `std::pair<_T1, _T2 >::second`.

**4.11.4.178** `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 663 of file `stl_multiset.h`.

**4.11.4.179** `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 768 of file `stl_multimap.h`.

```
4.11.4.180 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 837 of file `stl_map.h`.

```
4.11.4.181 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 1483 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

**4.11.4.182** `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 1827 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::begin()`, `std::deque<_Tp, _Alloc >::end()`, `equal()`, and `std::deque<_Tp, _Alloc >::size()`.

**4.11.4.183** `template<typename _StateT > bool std::operator==(const fpos<_StateT > & __lhs, const fpos<_StateT > & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 216 of file `postypes.h`.

**4.11.4.184** `template<typename _Tp, typename _Alloc > bool std::operator==(const forward_list<_Tp, _Alloc > & __lx, const forward_list<_Tp, _Alloc > & __ly) [inline]`

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 377 of file `forward_list.tcc`.

References `std::forward_list<_Tp, _Alloc >::cbegin()`, and `std::forward_list<_Tp, _Alloc >::cend()`.

**4.11.4.185** `template<typename _Tp, typename _Alloc > bool std::operator==(const _Fwd_list_iterator<_Tp, _Alloc > & __x, const _Fwd_list_const_iterator<_Tp, _Alloc > & __y) [inline]`

Forward [list iterator](#) equality comparison.

Definition at line 237 of file `forward_list.h`.

**4.11.4.186** `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 2366 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`.

**4.11.4.187** `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 2354 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**4.11.4.188** `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 2333 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**4.11.4.189** `template<typename _Res , typename... _Args> bool  
std::operator==( _M_clear_type *, 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 2176 of file `functional`.

**4.11.4.190** `template<typename _Res , typename... _Args> bool  
std::operator==(const function<_Res(_Args...)> & __f,  
_M_clear_type *) [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 2170 of file functional.

**4.11.4.191** `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 1202 of file stl\_vector.h.

**4.11.4.192** `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 stl\_stack.h.

**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]`

Returns  $y < x$ .

Definition at line 707 of file stl\_set.h.

**4.11.4.194** `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 300 of file stl\_queue.h.

**4.11.4.195** `template<class _T1 , class _T2 > bool std::operator<(const pair<  
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses operator< to find the result.

Definition at line 177 of file stl\_pair.h.

**4.11.4.196** `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 694 of file `stl_multiset.h`.

**4.11.4.197** `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 799 of file `stl_multimap.h`.

**4.11.4.198** `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 868 of file `stl_map.h`.

**4.11.4.199** `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 1525 of file `stl_list.h`.

**4.11.4.200** `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 1860 of file `stl_deque.h`.

**4.11.4.201** `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 1254 of file forward\_list.h.

**4.11.4.202** `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 2477 of file basic\_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.203** `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 2465 of file basic\_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.204** `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 2453 of file basic\_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.205** `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 1214 of file stl\_vector.h.

**4.11.4.206** `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.

**4.11.4.207** `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 721 of file stl\_set.h.

**4.11.4.208** `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 312 of file `stl_queue.h`.

**4.11.4.209** `template<class _T1, class _T2 > bool std::operator>= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 189 of file `stl_pair.h`.

**4.11.4.210** `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 708 of file `stl_multiset.h`.

**4.11.4.211** `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 813 of file `stl_multimap.h`.

**4.11.4.212** `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 882 of file `stl_map.h`.

**4.11.4.213** `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 1537 of file stl\_list.h.

**4.11.4.214** `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 1874 of file stl\_deque.h.

**4.11.4.215** `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 1261 of file forward\_list.h.

**4.11.4.216** `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 2551 of file basic\_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.



**4.11.4.217** `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 2539 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.218** `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 2527 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.219** `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) [inline]`

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 547 of file `vstring.tcc`.

**4.11.4.220** `template<typename _CharT , typename _Traits , typename  
_Alloc > basic_istream< _CharT, _Traits > & std::operator>>  
(basic_istream< _CharT, _Traits > & __is, basic_string< _CharT,  
_Traits, _Alloc > & __str) [inline]`

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 996 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()`.

**4.11.4.221** `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 847 of file istream.

**4.11.4.222** `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 756 of file `istream`.

**4.11.4.223** `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 751 of file `istream`.

**4.11.4.224** `template<> basic_istream<char>& std::operator>>  
(basic_istream< char > & __in, 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.

**4.11.4.225** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::operator>>(basic_istream< _CharT,  
_Traits > & __in, _CharT * __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 935 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()`.

**4.11.4.226** `template<class _Traits> basic_istream<char, _Traits>& std::operator>>(basic_istream<char, _Traits> & __in, signed char & __c) [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 709 of file `istream`.

```
4.11.4.227 template<class _Traits > basic_istream<char, _Traits>&
std::operator>>(basic_istream< char, _Traits > & __in, unsigned
char & __c) [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 704 of file `istream`.

**4.11.4.228** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::operator>> (basic_istream<_CharT, _Traits > & __in, _CharT & __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 903 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()`.



**4.11.4.229** `template<class _CharT, class _Traits, size_t _Nb>  
std::basic_istream<_CharT, _Traits>& std::operator>>  
(std::basic_istream<_CharT, _Traits> & __is, bitset<_Nb> &  
__x) [inline]`

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 1403 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()`.

**4.11.4.230** `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 1384 of file `bitset`.

**4.11.4.231** `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 1375 of file `bitset`.

**4.11.4.232** `template<typename _InputIterator , typename _OutputIterator ,  
typename _BinaryOperation > _OutputIterator std::partial_sum  
(_InputIterator __first, _InputIterator __last, _OutputIterator  
__result, _BinaryOperation __binary_op) [inline]`

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 273 of file `stl_numeric.h`.

**4.11.4.233** `template<typename _InputIterator , typename _OutputIterator  
> _OutputIterator std::partial_sum (_InputIterator __first,  
_InputIterator __last, _OutputIterator __result) [inline]`

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 233 of file `stl_numeric.h`.

**4.11.4.234** `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 292 of file `iomanipl`.

**4.11.4.235** `template<typename _Tp > reference_wrapper<_Tp> std::ref  
(reference_wrapper<_Tp > __t) [inline]`

Partial specialization.

Definition at line 451 of file `functional`.

References `ref()`.

**4.11.4.236** `template<typename _Tp > reference_wrapper<_Tp> std::ref (_Tp  
& __t) [inline]`

Denotes a reference should be taken to a variable.

Definition at line 439 of file `functional`.

Referenced by `ref()`.

**4.11.4.237** `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) [inline]`

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 3812 of file `stl_algo.h`.

**4.11.4.238** `_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 63 of file `iomanip`.

**4.11.4.239** `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 112 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

**4.11.4.240** `ios_base& std::right (ios_base & __base)` `[inline]`

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 923 of file ios\_base.h.

References `std::ios_base::setf()`.

#### 4.11.4.241 `ios_base& std::scientific (ios_base & __base) [inline]`

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 965 of file ios\_base.h.

References `std::ios_base::setf()`.

Referenced by `operator<<()`.

#### 4.11.4.242 `new_handler std::set_new_handler (new_handler) throw ()`

Takes a replacement handler as the argument, returns the previous handler.

#### 4.11.4.243 `_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 124 of file iomanip.

#### 4.11.4.244 `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 162 of file iomanip.

**4.11.4.245** `_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 93 of file `iomanip`.

**4.11.4.246** `_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 192 of file `iomanip`.

**4.11.4.247** `_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 222 of file `iomanip`.

**4.11.4.248** `ios_base& std::showbase (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 810 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowbase()`.

**4.11.4.249 ios\_base& std::showpoint (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::showpoint).

Definition at line 826 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by noshowpoint().

**4.11.4.250 ios\_base& std::showpos (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::showpos).

Definition at line 842 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by noshowpos().

**4.11.4.251 ios\_base& std::skipws (ios\_base & \_\_base) [inline]**

Calls base.setf(ios\_base::skipws).

Definition at line 858 of file ios\_base.h.

References std::ios\_base::setf().

Referenced by noskipws(), operator>>(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

**4.11.4.252 template<typename \_Tp, typename \_Alloc > void std::swap (vector<\_Tp, \_Alloc > & \_\_x, vector<\_Tp, \_Alloc > & \_\_y) [inline]**

See [std::vector::swap\(\)](#).

Definition at line 1220 of file stl\_vector.h.

References std::vector<\_Tp, \_Alloc >::swap().

**4.11.4.253 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 728 of file stl\_set.h.

References std::set<\_Key, \_Compare, \_Alloc >::swap().

**4.11.4.254** `template<class _T1 , class _T2 > void std::swap (pair< _T1, _T2 > & __x, pair< _T1, _T2 > & __y) [inline]`

See `std::pair::swap()`.

Definition at line 198 of file `stl_pair.h`.

**4.11.4.255** `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 715 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::swap()`.

**4.11.4.256** `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 820 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

**4.11.4.257** `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 889 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::swap()`.

**4.11.4.258** `template<typename _Tp , typename _Alloc > void std::swap (list< _Tp, _Alloc > & __x, list< _Tp, _Alloc > & __y) [inline]`

See `std::list::swap()`.

Definition at line 1543 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::swap()`.



**4.11.4.259** `template<typename _Tp, typename _Alloc > void std::swap  
(deque< _Tp, _Alloc > & __x, deque< _Tp, _Alloc > & __y)  
[inline]`

See [std::deque::swap\(\)](#).

Definition at line 1881 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::swap()`.

**4.11.4.260** `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 1275 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::swap()`.

**4.11.4.261** `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 2564 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

**4.11.4.262** `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 2206 of file `functional`.

Referenced by `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert()`, `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert()`.

min\_insert(), \_\_gnu\_parallel::\_\_parallel\_nth\_element(), \_\_gnu\_parallel::\_\_parallel\_partition(), \_\_gnu\_parallel::\_\_qsb\_divide(), \_\_gnu\_parallel::\_\_qsb\_local\_sort\_with\_helping(), \_\_rotate(), \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base >::assign(), std::basic\_string<char >::assign(), std::regex\_traits<\_Ch\_type >::imbue(), \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base >::operator=(), std::vector<result\_type >::operator=(), std::set<\_Key, \_Compare, \_Alloc >::operator=(), std::multiset<\_Key, \_Compare, \_Alloc >::operator=(), std::multimap<\_Key, \_Tp, \_Compare, \_Alloc >::operator=(), std::map<\_Key, \_Tp, \_Compare, \_Alloc >::operator=(), std::list<\_Tp, \_Alloc >::operator=(), std::forward\_list<\_Tp, \_Alloc >::operator=(), std::basic\_string<char >::operator=(), std::vector<result\_type >::swap(), std::list<\_Tp, \_Alloc >::swap(), std::function<\_Res(\_ArgTypes...) >::swap(), and std::forward\_list<\_Tp, \_Alloc >::swap().

#### 4.11.4.263 `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()`.

#### 4.11.4.264 `template<typename _CharT > _CharT std::toupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.toupper(__c)`.

#### 4.11.4.265 `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 106 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

**4.11.4.266** `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 531 of file `stl_uninitialized.h`.

References `__iterator_category()`.

**4.11.4.267** `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 163 of file `stl_uninitialized.h`.

**4.11.4.268** `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 279 of file `stl_uninitialized.h`.

**4.11.4.269 ios\_base& std::unitbuf (ios\_base & \_\_base) [inline]**

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 890 of file `ios_base.h`.

References `std::ios_base::setf()`.

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

**4.11.4.270 ios\_base& std::uppercase (ios\_base & \_\_base) [inline]**

Calls `base.setf(ios_base::uppercase)`.

Definition at line 874 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nouppercase()`.

**4.11.4.271 template<typename \_Facet > const \_Facet & std::use\_facet (const locale & \_\_loc) [inline]**

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 105 of file `locale_classes.tcc`.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, `std::regex_traits<_Ch_type>::transform()`, and `std::regex_traits<_Ch_type>::translate_nocase()`.

#### 4.11.4.272 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::ws (basic_istream<_CharT, _Traits> & __is) [inline]`

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 996 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 `ostream std::cerr`

Linked to standard error (unbuffered).

#### 4.11.5.2 `istream std::cin`

Linked to standard input.

**4.11.5.3 ostream std::clog**

Linked to standard error (buffered).

**4.11.5.4 ostream std::cout**

Linked to standard output.

**4.11.5.5 wostream std::wcerr**

Linked to standard error (unbuffered).

**4.11.5.6 wistream std::wcin**

Linked to standard input.

**4.11.5.7 wostream std::wclog**

Linked to standard error (buffered).

**4.11.5.8 wostream std::wcout**

Linked to standard output.

## 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 [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 _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 _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 _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 _Tp , typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__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 _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 _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 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 _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__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 _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 _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 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 _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 _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 _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 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 _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 _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 _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 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 _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 _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 _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 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 _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__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 _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `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 _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `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 (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `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 _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 _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 _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _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.

## 4.13 std::\_\_detail Namespace Reference

Implementation details not part of the namespace `std` interface.

### Functions

- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __-`  
`first, _Iterator __last)`
- `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, std::input_iterator_tag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter __lower_bound (_RAIter __first, _RAIter __last, const _Tp &__val)`
- `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)`
- `template<typename _Value, bool __cache >`  
`bool operator== (const _Node_iterator_base< _Value, __cache > &__x, const`  
`_Node_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 _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 _Tp, typename _BinaryOper, typename _Tag > _Tp accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _Iter, typename _Tp, typename _Tag > _Tp accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `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 _BinaryOperation, typename _IteratorTag > _Tp accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag > _Tp accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `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 _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 > _OIter adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `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 _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __adjacent_difference_switch (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _BiPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_FIter __adjacent_find_switch (_FIter, _FIter, _BiPredicate, _IterTag)`
- `template<typename _FIter, typename _IterTag >`  
`_FIter __adjacent_find_switch (_FIter, _FIter, _IterTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _FIterator, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _RAIter >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _IIter, typename _Predicate, typename _IterTag >`  
`iterator\_traits< _IIter >::difference_type __count_if_switch (_IIter, _IIter, _Predicate, _IterTag)`
- `template<typename _IIter, typename _Predicate, typename _IteratorTag >`  
`iterator\_traits< _IIter >::difference_type __count_if_switch (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)`
- `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 _IIter, typename _Tp, typename _IterTag >`  
`iterator\_traits< _IIter >::difference_type __count_switch (_IIter, _IIter, const _Tp &, _IterTag)`
- `template<typename _IIter, typename _Tp, typename _IteratorTag >`  
`iterator\_traits< _IIter >::difference_type __count_switch (_IIter __begin, _IIter __end, const _Tp &__value, _IteratorTag)`
- `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 _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random\_access\_iterator\_tag, _IterTag)`
- `template<typename _Iter, typename _FIter, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `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 _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 _IteratorTag1, typename _IteratorTag2 >`  
`>`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `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 _IteratorTag >`  
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `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 _IteratorTag >`  
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function __for_each_switch (_Iter, _Iter, _Function, _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 _OIter, typename _Size, typename _Generator, typename _IterTag > _OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
  - `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 _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag > _OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
  - `template<typename _Filter, typename _Generator, typename _IterTag > void __generate_switch (_Filter, _Filter, _Generator, _IterTag)`
  - `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 _Filter, typename _Generator, typename _IteratorTag > void __generate_switch (_Filter __begin, _Filter __end, _Generator __gen, _IteratorTag)`
  - `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, _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, 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 _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 _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool __lexicographical_compare_switch (_IIter1, _IIter1, _IIter2, _IIter2, _-`  
`Predicate, _IterTag1, _IterTag2)`
- `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 _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool __lexicographical_compare_switch (_IIter1 __begin1, _IIter1 __end1,`  
`_IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _-`  
`IteratorTag2)`
- `template<typename _FIter, typename _Compare, typename _IterTag >`  
`_FIter __max_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `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 _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _-`  
`Compare __comp, _IteratorTag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _-`  
`Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_`  
`access\_iterator\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _-`  
`IterTag1, _IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 _-`  
`begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random\_`  
`access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2`  
`__begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _-`  
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _FIter, typename _Compare, typename _IterTag >`  
`_FIter __min_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `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 _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _-`  
`Compare __comp, _IteratorTag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`pair< _IIter1, _IIter2 > __mismatch_switch (_IIter1, _IIter1, _IIter2, _-`  
`Predicate, _IterTag1, _IterTag2)`
- `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 _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1 ,`  
`typename _IteratorTag2 >`  
`pair< _IIter1, _IIter2 > __mismatch_switch (_IIter1 __begin1, _IIter1 __end1,`  
`_IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, random\_`  
`access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`  
`_Tag2 >`  
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _-`  
`Tag2)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __partial_sum_switch (_IIter __begin, _IIter __end, _-`  
`OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_`  
`tag, random\_access\_iterator\_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename`  
`_IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __partial_sum_switch (_IIter __begin, _IIter __end, _-`  
`OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _-`  
`IteratorTag2)`
- `template<typename _FIter, typename _Predicate, typename _IterTag >`  
`_FIter __partition_switch (_FIter, _FIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __-`  
`pred, random\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator __partition_switch (_FIterator __begin, _FIterator __end, _Predicate`  
`__pred, _IteratorTag)`
- `template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void __replace_if_switch (_FIter, _FIter, _Predicate, const _Tp &, _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 _Tp, typename _IterTag >`  
`void __replace_switch (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)`
- `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 _FIterator, typename _Tp, typename _IteratorTag >`  
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `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 _BiPredicate >`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, 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 _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 _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _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 _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)`
- `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 _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 _IteratorTag1 , typename _IteratorTag2 >`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `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 _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OIter __set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _OutputRAIter , typename _Predicate >`  
`_OutputRAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random\_access\_iterator\_tag, 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 _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OIter __set_intersection_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _OutputRAIter , typename _Predicate >`  
`_OutputRAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random\_access\_iterator\_tag, 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_intersection_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OIter __set_symmetric_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _OutputRAIter , typename _Predicate >`  
`Predicate >`

- `_Output_RAlter __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 __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_union_switch` (`_Iter1`, `_Iter1`, `_Iter2`, `_Iter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter __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 _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 _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 _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter __transform1_switch` (`_Iter`, `_Iter`, `_OIter`, `_UnaryOperation`, `_IterTag1`, `_IterTag2`)
  - `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 _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 _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 _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 _IIter1 , typename _IIter2 , typename _OutputIterator , typename _`  
`BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OutputIterator __transform2_switch (_IIter1 __begin1, _IIter1 __end1, _`  
`IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _`  
`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 _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 _IIter , typename _OIter , typename _Predicate , typename _IterTag1 , typename`  
`_IterTag2 >`  
`_OIter __unique_copy_switch (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _`  
`IIterTag2)`
- `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 _IIter , typename _OutputIterator , typename _Predicate , typename _`  
`IteratorTag1 , typename _IteratorTag2 >`  
`_OutputIterator __unique_copy_switch (_IIter __begin, _IIter __last, _`  
`OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter , typename _Tp , typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu\_parallel::`  
`Parallelism)`
- `template<typename _IIter , typename _Tp , typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_`  
`tag)`
- `template<typename _IIter , typename _Tp , typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper)`
- `template<typename _IIter , typename _Tp , typename _BinaryOperation >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation`  
`__binary_op)`

- `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 >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_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 >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_gnu\_-`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_-`  
`parallel::Parallelism)`
- `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::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_-`  
`tag)`
- `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)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, _BinaryOperation __binary_op)`
- `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 >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_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::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_`  
`tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _`  
`BinaryPredicate __pred)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `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, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`  
`const _Tp &__value)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`  
`_Predicate __pred)`
- `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, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `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 >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
  
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `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, 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 >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `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, _Iter, _Function)`

- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::\_\_Parallelism)`
- `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)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, \_\_gnu\_parallel::\_\_Parallelism)`
- `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::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 _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, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `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 _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2, _Iter2 __end2, _Predicate __pred)`
- `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, \_\_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 _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter)`

- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_`  
`parallel::Parallelism __parallelism_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::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _`  
`IIter2 __end2, _OutputIterator __result)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _`  
`IIter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __result, _Compare __comp, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _`  
`IIter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_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::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`

- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_IIter, _IIter, _OIter, _BinaryOper)`
- `template<typename _IIter, typename _OIter >`  
`_OIter partial_sum (_IIter, _IIter, _OIter __result)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_IIter, _IIter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter >`  
`_OIter partial_sum (_IIter, _IIter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _IIter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `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)`

- `template<typename _FIterator, typename _Predicate >`  
`_FIterator partition (_FIterator __begin, _FIterator __end, _Predicate __pred,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand)`
- `template<typename _RAIter >`  
`void random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, __gnu-`  
`parallel::sequential_tag)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_-`  
`Parallelism)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu-`  
`parallel::sequential_tag)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _FIterator, typename _Tp >`  
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`  
`const _Tp &__new_value)`
- `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 _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_-`  
`Parallelism)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, __gnu-`  
`parallel::sequential_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 _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 _FIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`  
`_Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, \_\_gnu\_parallel::sequential\_`  
`tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __`  
`begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `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 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __`  
`begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_FIter search_n (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)`
- `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 _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count,`  
`const _Tp &__val, _BinaryPredicate __binary_pred)`
- `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, \_\_gnu\_`  
`parallel::sequential\_tag)`

- `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 _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator 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 set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __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, \_\_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, typename _Predicate > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `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 _OIter > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_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 _OutputIterator > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `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 > _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, typename _Predicate > _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `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 > _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_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 _OutputIterator > _OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `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 _OutputIterator > _OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare > void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter > void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter > void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_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\_sampling\_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::parallel\_tag __parallelism)`
- `template<typename _RAIter > void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter > void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism > void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __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::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quick\_sort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::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::parallel\_tag __parallelism)`
- `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)`
- `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::sequential\_tag)`
- `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 _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::\_Parallelism)`
- `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 _Iter1, typename _Iter2, typename _OutputIterator, typename _`  
`BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_OutputIterator __result, _BinaryOperation __binary_op)`
- `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 _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 _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __`  
`result, _UnaryOperation __unary_op)`
- `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 _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 _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `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)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out, \_\_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 [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 _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 _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 _Iterator, typename _Sequence >`  
`bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`  
`Alloc > &__rhs)`
- `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 _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 _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 _Iterator, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__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 _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 _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _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 _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 _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _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 _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 _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const \_\_iterator\_tracker< _Iterator, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const \_\_iterator\_tracker< _IteratorL, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _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 _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 _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const \_\_iterator\_tracker< _Iterator, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const \_\_iterator\_tracker< _IteratorL, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _IteratorR, _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 list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`  
`> &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__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 _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `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 _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `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 (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 (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `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 _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 _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 _Tp, typename _Alloc >`  
`void swap (list<_Tp, _Alloc > &__lhs, list<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque<_Tp, _Alloc > &__lhs, deque<_Tp, _Alloc > &__rhs)`

#### 4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

## 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 \_ToDuration , typename \_Rep , typename \_Period >  
`enable_if`< \_\_is\_duration< \_ToDuration >::value, \_ToDuration >::type  
`duration_cast` (const `duration`< \_Rep, \_Period > &\_d)
- template<typename \_Clock , typename \_Duration1 , typename \_Duration2 >  
bool **operator!=** (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const  
`time_point`< \_Clock, \_Duration2 > &\_rhs)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`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 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 _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 _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 _Period1, typename _Clock, typename _Duration2 >`  
`time\_point< _Clock, typename common_type< duration< _Rep1, _Period1 >,`   
`_Duration2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs,`  
`const time\_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`  
`time\_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`  
`_Period2 > >::type > operator+ (const time\_point< _Clock, _Duration1 >`  
`&__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 dura-`  
`tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`common_type< _Duration1, _Duration2 >::type operator- (const time\_point<`  
`_Clock, _Duration1 > &__lhs, const time\_point< _Clock, _Duration2 > &__-`  
`rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`  
`time\_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`  
`_Period2 > >::type > operator- (const time\_point< _Clock, _Duration1 > &__-`  
`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 dura-`  
`tion< _Rep2, _Period2 > &__rhs)`
- `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 _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 _Clock, typename _Duration1, typename _Duration2 >`  
**bool operator**< (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const `time_`-  
`point`< \_Clock, \_Duration2 > &\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
**bool operator**< (const `duration`< \_Rep1, \_Period1 > &\_lhs, const `duration`<  
\_Rep2, \_Period2 > &\_rhs)
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
**bool operator**<= (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const  
`time_point`< \_Clock, \_Duration2 > &\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
**bool operator**<= (const `duration`< \_Rep1, \_Period1 > &\_lhs, const `duration`<  
\_Rep2, \_Period2 > &\_rhs)
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
**bool operator**== (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const  
`time_point`< \_Clock, \_Duration2 > &\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
**bool operator**== (const `duration`< \_Rep1, \_Period1 > &\_lhs, const `duration`<  
\_Rep2, \_Period2 > &\_rhs)
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
**bool operator**> (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const `time_`-  
`point`< \_Clock, \_Duration2 > &\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
**bool operator**> (const `duration`< \_Rep1, \_Period1 > &\_lhs, const `duration`<  
\_Rep2, \_Period2 > &\_rhs)
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
**bool operator**>= (const `time_point`< \_Clock, \_Duration1 > &\_lhs, const  
`time_point`< \_Clock, \_Duration2 > &\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
**bool operator**>= (const `duration`< \_Rep1, \_Period1 > &\_lhs, const `duration`<  
\_Rep2, \_Period2 > &\_rhs)
- `template<typename _ToDuration, typename _Clock, typename _Duration >`  
`enable_if`< \_\_is\_duration< \_ToDuration >::value, `time_point`< \_Clock, \_-  
ToDuration > >::type `time_point_cast` (const `time_point`< \_Clock, \_Duration  
> &\_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 488 of file chrono.

### 4.16.2.2 `typedef duration<int64_t, micro> std::chrono::microseconds`

microseconds

Definition at line 476 of file chrono.

### 4.16.2.3 `typedef duration<int64_t, milli> std::chrono::milliseconds`

milliseconds

Definition at line 479 of file chrono.

### 4.16.2.4 `typedef duration<int, ratio< 60> > std::chrono::minutes`

minutes

Definition at line 485 of file chrono.

### 4.16.2.5 `typedef duration<int64_t, nano> std::chrono::nanoseconds`

nanoseconds

Definition at line 473 of file chrono.

### 4.16.2.6 `typedef duration<int64_t > std::chrono::seconds`

seconds

Definition at line 482 of file chrono.

### 4.16.3 Function Documentation

**4.16.3.1** `template<typename _ToDuration , typename _Rep , typename _Period > enable_if<__is_duration<_ToDuration>::value, _ToDuration>::type std::chrono::duration_cast (const duration<_Rep, _Period > & __d) [inline]`

duration\_cast

Definition at line 155 of file chrono.

Referenced by std::this\_thread::sleep\_for(), and time\_point\_cast().

**4.16.3.2** `template<typename _ToDuration , typename _Clock , typename _Duration > enable_if<__is_duration<_ToDuration>::value, time_point<_Clock, _ToDuration> >::type std::chrono::time_point_cast (const time_point<_Clock, _Duration > & __t) [inline]`

time\_point\_cast

Definition at line 550 of file chrono.

References duration\_cast().

## 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** ([decimal128 \\_\\_d](#))
- double **decimal\_to\_double** ([decimal64 \\_\\_d](#))
- double **decimal\_to\_double** ([decimal32 \\_\\_d](#))
- float **decimal\_to\_float** ([decimal128 \\_\\_d](#))
- float **decimal\_to\_float** ([decimal64 \\_\\_d](#))
- float **decimal\_to\_float** ([decimal32 \\_\\_d](#))
- long double **decimal\_to\_long\_double** ([decimal128 \\_\\_d](#))
- long double **decimal\_to\_long\_double** ([decimal64 \\_\\_d](#))
- long double **decimal\_to\_long\_double** ([decimal32 \\_\\_d](#))
- long long **decimal\_to\_long\_long** ([decimal128 \\_\\_d](#))
- long long **decimal\_to\_long\_long** ([decimal64 \\_\\_d](#))
- long long **decimal\_to\_long\_long** ([decimal32 \\_\\_d](#))

- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (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!=** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)

- bool **operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)

- **decimal32 operator\*** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator\*** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator\*** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, int \_\_rhs)
- **decimal32 operator\*** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator+** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator+** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator+** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal64 operator+** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** (int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, int \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator+** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal32 operator+** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)

- **decimal32 operator+** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator+** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator+** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator+** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator+** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, int \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator+** ([decimal128](#) \_\_rhs)
- **decimal64 operator+** ([decimal64](#) \_\_rhs)
- **decimal32 operator+** ([decimal32](#) \_\_rhs)
- **decimal128 operator-** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator-** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator-** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal64 operator-** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** (int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)



- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), [int \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal32 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal32 operator-** ([unsigned long long \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([long long \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([unsigned long \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([long \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([unsigned int \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([int \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [unsigned long long \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [long long \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [unsigned long \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [long \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [unsigned int \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [int \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal128 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_rhs](#))
- **decimal128 operator/** ([unsigned long long \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([long long \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([unsigned long \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([long \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([unsigned int \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([int \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [unsigned long long \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [long long \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [unsigned long \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [long \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [unsigned int \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [int \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal128 operator/** ([decimal128 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal128 operator/** ([decimal64 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator/** ([decimal32 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal64 operator/** ([unsigned long long \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator/** ([long long \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator/** ([unsigned long \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator/** ([long \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator/** ([unsigned int \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator/** ([int \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))

- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, int \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal32 operator/** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, int \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** (int \_\_lhs, [decimal64](#) \_\_rhs)

- bool **operator**< (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal64 \_\_rhs)

- bool **operator==** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** (long long \_\_lhs, [decimal64](#) \_\_rhs)

- bool **operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)

- bool **operator**>= (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal32 \_\_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.

### 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.

### 5.2 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` `format_default`
- static const `match_flag_type` `format_first_only`
- static const `match_flag_type` `format_no_copy`
- static const `match_flag_type` `format_sed`
- static const `match_flag_type` `match_any`
- static const `match_flag_type` `match_continuous`
- static const `match_flag_type` `match_default`
- static const `match_flag_type` `match_not_bol`
- static const `match_flag_type` `match_not_bow`
- static const `match_flag_type` `match_not_eol`
- static const `match_flag_type` `match_not_eow`
- static const `match_flag_type` `match_not_null`
- static const `match_flag_type` `match_prev_avail`

### 5.1 Regular Expression Syntax Options

- enum `__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` `syntax_option_type`
- static const `syntax_option_type` `awk`
- static const `syntax_option_type` `basic`
- static const `syntax_option_type` `collate`
- static const `syntax_option_type` `ECMAScript`



- static const [syntax\\_option\\_type](#) `egrep`
- static const [syntax\\_option\\_type](#) `extended`
- static const [syntax\\_option\\_type](#) `grep`
- static const [syntax\\_option\\_type](#) `icase`
- static const [syntax\\_option\\_type](#) `nosubs`
- static const [syntax\\_option\\_type](#) `optimize`

### 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\\_backref](#) ([\\_S\\_error\\_backref](#))
  - static const [error\\_type](#) [error\\_badbrace](#) ([\\_S\\_error\\_badbrace](#))
  - static const [error\\_type](#) [error\\_badrepeat](#) ([\\_S\\_error\\_badrepeat](#))
  - static const [error\\_type](#) [error\\_brace](#) ([\\_S\\_error\\_brace](#))
  - static const [error\\_type](#) [error\\_brack](#) ([\\_S\\_error\\_brack](#))
  - static const [error\\_type](#) [error\\_collate](#) ([\\_S\\_error\\_collate](#))
  - static const [error\\_type](#) [error\\_complexity](#) ([\\_S\\_error\\_complexity](#))
  - static const [error\\_type](#) [error\\_ctype](#) ([\\_S\\_error\\_ctype](#))
  - static const [error\\_type](#) [error\\_escape](#) ([\\_S\\_error\\_escape](#))
  - static const [error\\_type](#) [error\\_paren](#) ([\\_S\\_error\\_paren](#))
  - static const [error\\_type](#) [error\\_range](#) ([\\_S\\_error\\_range](#))
  - static const [error\\_type](#) [error\\_space](#) ([\\_S\\_error\\_space](#))
  - 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 198 of file `tr1_impl/regex`.

#### 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 78 of file `tr1_impl/regex`.

### 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 175 of file `tr1_impl/regex`.

#### 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 52 of file `tr1_impl/regex`.

#### 4.19.3.3 `enum std::regex_constants::error_type`

The expression contained an invalid back reference.

Definition at line 310 of file `tr1_impl/regex`.

### 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 144 of file tr1\_impl/regex.

**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 124 of file tr1\_impl/regex.

**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 105 of file tr1\_impl/regex.

**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 [tr1](#) section [7.13]. 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 115 of file tr1\_impl/regex.

**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 160 of file tr1\_impl/regex.

**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 132 of file tr1\_impl/regex.

**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 <= 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 <= 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 280 of file `tr1_impl/regex`.

#### 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 301 of file `tr1_impl/regex`.

#### 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 295 of file `tr1_impl/regex`.

#### 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 288 of file `tr1_impl/regex`.

**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 152 of file `tr1_impl/regex`.

**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 84 of file `tr1_impl/regex`.

**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 235 of file `tr1_impl/regex`.

**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 245 of file `tr1_impl/regex`.

**4.19.5.15 const match\_flag\_type std::regex\_constants::match\_default [static]**

The default matching rules.

Definition at line 203 of file `tr1_impl/regex`.

**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 210 of file `tr1_impl/regex`.

**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 223 of file `tr1_impl/regex`.

**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 `(\)` in the regular expression shall not match `[last, last)`.

Definition at line 217 of file `tr1_impl/regex`.

**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 229 of file `tr1_impl/regex`.

**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 240 of file `tr1_impl/regex`.

**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 252 of file `tr1_impl/regex`.

**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 91 of file `tr1_impl/regex`.



**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 99 of file `tr1_impl/regex`.

## 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 85 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:**

$x \leq y$

This function uses  $<$  to determine its result.

Definition at line 111 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 98 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  $\geq$  for arbitrary types, in terms of  $<$ .

**Parameters:**

- x* A thing.
- y* Another thing.

**Returns:**

$x \geq y$

This function uses  $<$  to determine its result.

Definition at line 124 of file stl\_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 252 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 271 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 265 of file `thread`.

References sleep\_for().

**4.21.2.4 void std::this\_thread::yield () [inline]**

yield

Definition at line 257 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 >`  
`__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 _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 \_Tp >  
std::complex< \_Tp > **conj** (const std::complex< \_Tp > &\_\_z)
- 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 >  
\_\_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 **hypergl** (\_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 >  
[std::complex](#)< \_Tp > **pow** (const [std::complex](#)< \_Tp > &\_\_x, const [std::complex](#)< \_Tp > &\_\_y)
- template<typename \_Tp >  
[std::complex](#)< \_Tp > **pow** (const \_Tp &\_\_x, const [std::complex](#)< \_Tp > &\_\_y)
- template<typename \_Tp >  
[std::complex](#)< \_Tp > **pow** (const [std::complex](#)< \_Tp > &\_\_x, const \_Tp &\_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **pow** (\_Tp \_\_x, \_Up \_\_y)
- long double **pow** (long double \_\_x, long double \_\_y)
- float **pow** (float \_\_x, float \_\_y)
- double **pow** (double \_\_x, double \_\_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.

# Chapter 5

## Class Documentation

### 5.1 `__atomic0::atomic_address` Struct Reference

29.4.2, address types

#### Public Member Functions

- `atomic_address` (void \*\_\_v)
- `atomic_address` (const [atomic\\_address](#) &)
- 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 \_\_m1, memory\_order \_\_m2)
- 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 \_\_m1, memory\_order \_\_m2)
- void \* `exchange` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)
- 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)
- bool `is_lock_free` () const
- void \* `load` (memory\_order \_\_m=memory\_order\_seq\_cst) const
- `operator void *` () const
- void \* `operator+=` (ptrdiff\_t \_\_d)
- void \* `operator-=` (ptrdiff\_t \_\_d)
- void \* `operator=` (void \*\_\_v)
- [atomic\\_address](#) & `operator=` (const [atomic\\_address](#) &) volatile
- void `store` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.1.1 Detailed Description

29.4.2, address types

Definition at line 103 of file atomic\_0.h.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.2 `__atomic0::atomic_bool` Struct Reference

[atomic\\_bool](#)

### Public Member Functions

- **atomic\_bool** (bool \_\_i)
- **atomic\_bool** (const [atomic\\_bool](#) &)
- 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 \_\_m1, memory\_order \_\_m2)
- 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 \_\_m1, memory\_order \_\_m2)
- bool **exchange** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const
- bool **load** (memory\_order \_\_m=memory\_order\_seq\_cst) const
- **operator bool** () const
- bool **operator=** (bool \_\_i)
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &) volatile
- void **store** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.2.1 Detailed Description

[atomic\\_bool](#)

Definition at line 391 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.3 `__atomic0::atomic_flag` Struct Reference

[atomic\\_flag](#)

Inherits `__atomic_flag_base`.

### Public Member Functions

- `atomic_flag` (bool \_\_i)
- `atomic_flag` (const [atomic\\_flag](#) &)
- void `clear` (memory\_order \_\_m=memory\_order\_seq\_cst)
- [atomic\\_flag](#) & `operator=` (const [atomic\\_flag](#) &) volatile
- bool `test_and_set` (memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.3.1 Detailed Description

[atomic\\_flag](#)

Definition at line 85 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.4 `__atomic2::atomic_address` Struct Reference

29.4.2, address types

### Public Member Functions

- `atomic_address` (void \*\_\_v)
- `atomic_address` (const [atomic\\_address](#) &)
- 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 \_\_m1, memory\_order \_\_m2)
- 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 \_\_m1, memory\_order \_\_m2)
- void \* `exchange` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)
- 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)
- bool `is_lock_free` () const
- void \* `load` (memory\_order \_\_m=memory\_order\_seq\_cst) const
- `operator void *` () const
- void \* `operator+=` (ptrdiff\_t \_\_d)
- void \* `operator-=` (ptrdiff\_t \_\_d)
- void \* `operator=` (void \*\_\_v)
- [atomic\\_address](#) & `operator=` (const [atomic\\_address](#) &) volatile
- void `store` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.4.1 Detailed Description

29.4.2, address types

Definition at line 81 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.5 `__atomic2::atomic_bool` Struct Reference

[atomic\\_bool](#)

### Public Member Functions

- **atomic\_bool** (bool \_\_i)
- **atomic\_bool** (const [atomic\\_bool](#) &)
- 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 \_\_m1, memory\_order \_\_m2)
- 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 \_\_m1, memory\_order \_\_m2)
- bool **exchange** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const
- bool **load** (memory\_order \_\_m=memory\_order\_seq\_cst) const
- **operator bool** () const
- bool **operator=** (bool \_\_i)
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &) volatile
- void **store** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.5.1 Detailed Description

[atomic\\_bool](#)

Definition at line 391 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)



## 5.6 `__atomic2::atomic_flag` Struct Reference

[atomic\\_flag](#)

Inherits `__atomic_flag_base`.

### Public Member Functions

- `atomic_flag` (bool `__i`)
- `atomic_flag` (const [atomic\\_flag](#) &)
- `atomic_flag` & `operator=` (const [atomic\\_flag](#) &) volatile

### 5.6.1 Detailed Description

[atomic\\_flag](#)

Definition at line 47 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.7 `__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.

### 5.7.1 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 47 of file `cxxabi-forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi-forced.h](#)

**5.8 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference** 771

---

**5.8 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference**

Policy for shared `__pool` objects.

Inherits `__common_pool_base<_PoolTp, _Thread >`.

**5.8.1 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 456 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.9 `__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**
- 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 ()

#### 5.9.1 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`.

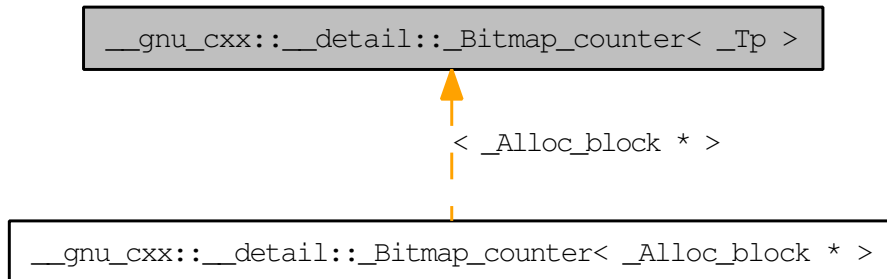
The documentation for this class was generated from the following file:

## 5.9 \_\_gnu\_cxx::\_\_detail::\_\_mini\_vector<\_Tp> Class Template Reference 773

- [bitmap\\_allocator.h](#)

## 5.10 `__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. 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 ()

### 5.10.1 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.

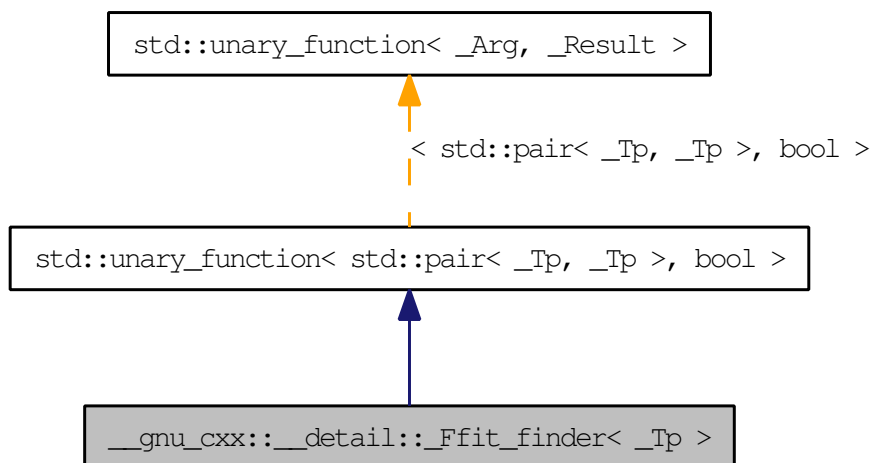
Definition at line 400 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.11 `__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 ()`

#### 5.11.1 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`.

## 5.11.2 Member Typedef Documentation

### 5.11.2.1 `typedef std::pair<_Tp, _Tp> std::unary_function< std::pair<_Tp, _Tp>, bool>::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.11.2.2 `typedef bool std::unary_function< std::pair<_Tp, _Tp>, bool>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

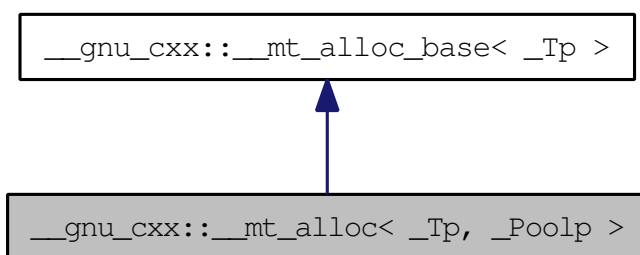


## 5.12 `__gnu_cxx::__mt_alloc< _Tp, _Poolp >` Class Template Reference

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

- `template<typename _Tp1, typename _Poolp1 >`  
`__mt_alloc` (`const __mt_alloc< _Tp1, _Poolp1 > &`) `throw ()`
- `__mt_alloc` (`const __mt_alloc &`) `throw ()`
- `const __pool_base::_Tune` **\_M\_get\_options** ()
- `void` **\_M\_set\_options** (`__pool_base::_Tune` `__t`)
- `const_pointer` **address** (`const_reference` `__x`) `const`
- `pointer` **address** (`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 ()

### 5.12.1 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>.

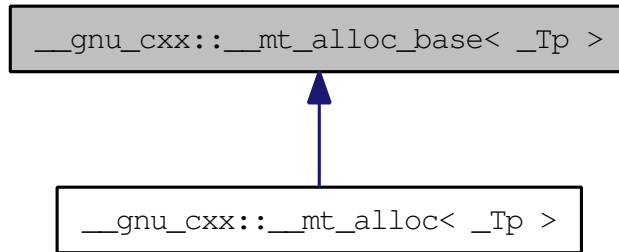
Definition at line 625 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.13 `__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

- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **address** (`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 ()`

#### 5.13.1 Detailed Description

```
template<typename _Tp> class __gnu_cxx::__mt_alloc_base< _Tp >
```

Base class for `_Tp` dependent member functions.

Definition at line 566 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

**5.14 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >` Struct  
Template Reference 781**

---

**5.14 `__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 >`.

**5.14.1 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.

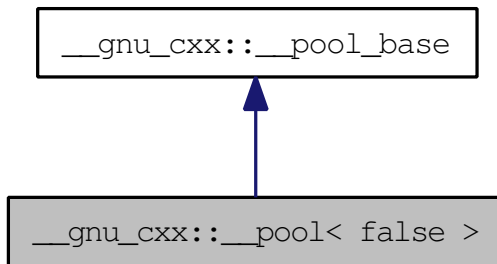
Definition at line 551 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.15 `__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`

### 5.15.1 Detailed Description

`template<> class __gnu_cxx::__pool<false>`

Specialization for single thread.

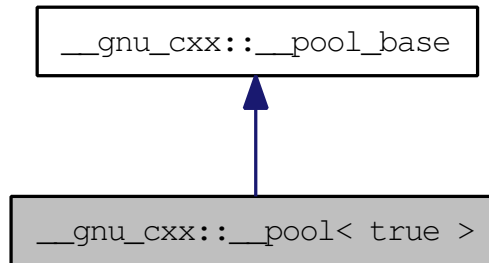
Definition at line 192 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.16 `__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`)
- `__attribute__` ((`__const__`)) void `_M_destroy_thread_key`(void \*) throw ()
- 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 ()
- `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`

### 5.16.1 Detailed Description

`template<> class __gnu_cxx::__pool< true >`

Specialization for thread enabled, via `gthreads.h`.

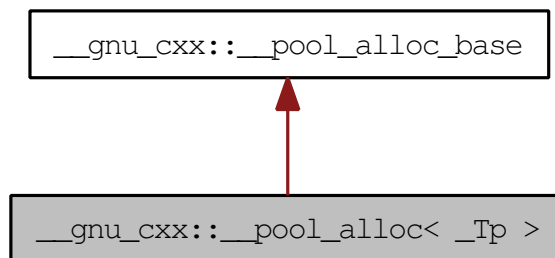
Definition at line 259 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.17 `__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

- `template<typename _Tp1 >`  
`__pool_alloc` (`const __pool_alloc< _Tp1 > &`) `throw ()`
- `__pool_alloc` (`const __pool_alloc &`) `throw ()`
- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **address** (`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

- `__attribute__((__const__))` `_Obj *volatile *_M_get_free_list(size_t __bytes)`  
throw ()
- `char *_M_allocate_chunk (size_t __n, int &__nobjs)`
- `__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`

### 5.17.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__pool_alloc<_Tp>`

Allocator using a memory pool with a single lock.

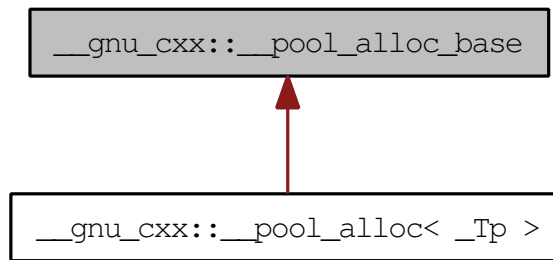
Definition at line 122 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 5.18 `__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

- `__attribute__((const)) _Obj *volatile *_M_get_free_list(size_t __bytes) throw ()`
- `char *_M_allocate_chunk (size_t __n, int &__nobjs)`
- `__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`

#### 5.18.1 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.

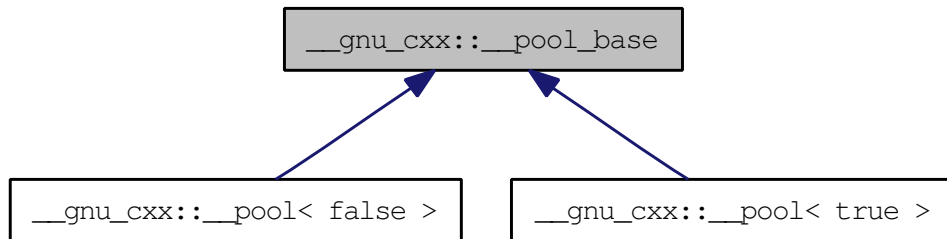
Definition at line 74 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 5.19 `__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`)
- `const _Tune & _M_get_options` () const
- `void _M_set_options` (`_Tune` `__t`)

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

#### 5.19.1 Detailed Description

Base class for pool object.

Definition at line 47 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.20 `__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

- `template<typename _InputIterator >`  
`__rc_string_base` (`_InputIterator __beg`, `_InputIterator __end`, `const _Alloc &__a`, `__a`)
- `__rc_string_base` (`size_type __n`, `_CharT __c`, `const _Alloc &__a`)
- `__rc_string_base` (`__rc_string_base &&__rcs`)
- `__rc_string_base` (`const __rc_string_base &__rcs`)
- `__rc_string_base` (`const _Alloc &__a`)
- `void _M_assign` (`const __rc_string_base &__rcs`)
- `size_type _M_capacity` () const
- `void _M_clear` ()
- `template<>`  
`bool _M_compare` (`const __rc_string_base &__rcs`) const
- `template<>`  
`bool _M_compare` (`const __rc_string_base &__rcs`) const
- `bool _M_compare` (`const __rc_string_base &`) const
- `_CharT * _M_data` () const
- `void _M_erase` (`size_type __pos`, `size_type __n`)
- `const allocator_type & _M_get_allocator` () const
- `allocator_type & _M_get_allocator` ()
- `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, const [\\_CharT](#) \* \_\_k1, const [\\_CharT](#) \* \_\_k2)
- static void **\_S\_copy\_chars** ([\\_CharT](#) \* \_\_p, [\\_CharT](#) \* \_\_k1, [\\_CharT](#) \* \_\_k2)
- static void **\_S\_copy\_chars** ([\\_CharT](#) \* \_\_p, [\\_\\_const\\_rc\\_iterator](#) \_\_k1, [\\_\\_const\\_rc\\_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)
- static void **\_S\_copy\_chars** ([\\_CharT](#) \* \_\_p, [\\_\\_sso\\_iterator](#) \_\_k1, [\\_\\_sso\\_iterator](#) \_\_k2)
- template<typename \_Iterator >  
static void **\_S\_copy\_chars** ([\\_CharT](#) \* \_\_p, \_Iterator \_\_k1, \_Iterator \_\_k2)
- static void **\_S\_move** ([\\_CharT](#) \* \_\_d, const [\\_CharT](#) \* \_\_s, size\_type \_\_n)



### 5.20.1 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:

```

 [_Rep]
 _M_length
[__rc_string_base<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_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 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 5.21 `__gnu_cxx::__scoped_lock` Class Reference

Scoped lock idiom.

### Public Types

- typedef `__mutex` `__mutex_type`

### Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

#### 5.21.1 Detailed Description

Scoped lock idiom.

Definition at line 241 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 5.22 `__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 `_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, __versa_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

- `template<class _InputIterator >`  
`__versa_string` (`_InputIterator` \_\_beg, `_InputIterator` \_\_end, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`size_type` \_\_n, `_CharT` \_\_c, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`const _CharT *` \_\_s, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`const _CharT *` \_\_s, `size_type` \_\_n, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n, `const _Alloc &` \_\_a)
- `__versa_string` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n=`npos`)
- `__versa_string` (`std::initializer_list< _CharT >` \_\_l, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`__versa_string &&` \_\_str)
- `__versa_string` (`const __versa_string &` \_\_str)
- `__versa_string` (`const _Alloc &` \_\_a)

- [\\_\\_versa\\_string](#) ()
- [~\\_\\_versa\\_string](#) ()
- [template<class \\_InputIterator >](#)  
[\\_\\_versa\\_string](#) & [append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string](#) & [append](#) (std::initializer\_list< \_CharT > \_\_l)
- [\\_\\_versa\\_string](#) & [append](#) (size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string](#) & [append](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & [append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string](#) & [assign](#) (std::initializer\_list< \_CharT > \_\_l)
- [template<class \\_InputIterator >](#)  
[\\_\\_versa\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string](#) & [assign](#) (size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & [assign](#) ([\\_\\_versa\\_string](#) &&\_\_str)
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- [reference](#) [at](#) (size\_type \_\_n)
- [const\\_reference](#) [at](#) (size\_type \_\_n) const
- [const\\_reference](#) [back](#) () const
- [reference](#) [back](#) ()
- [const\\_iterator](#) [begin](#) () const
- [iterator](#) [begin](#) ()
- [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, size\_type \_\_n2) const
- [int](#) [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- [int](#) [compare](#) (const \_CharT \*\_\_s) 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`
- `const_iterator end () const`
- `iterator end ()`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `__versa_string & erase (size_type __pos=0, size_type __n=npos)`
- `size_type find (_CharT __c, size_type __pos=0) const`
- `size_type find (const _CharT *__s, size_type __pos=0) const`
- `size_type find (const __versa_string &__str, size_type __pos=0) const`
- `size_type find (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=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 __versa_string &__str, 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=0) const`
- `size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) 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=npow) const`
- `size_type find_last_not_of (const _CharT *__s, size_type __pos=npow) 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=npow) const`
- `size_type find_last_of (_CharT __c, size_type __pos=npow) const`
- `size_type find_last_of (const _CharT *__s, size_type __pos=npow) const`
- `size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find_last_of (const __versa_string &__str, size_type __pos=npow) const`
- `const_reference front () const`
- `reference front ()`
- `allocator_type get_allocator () const`
- `iterator insert (iterator __p, _CharT __c)`
- `__versa_string & insert (size_type __pos, size_type __n, _CharT __c)`
- `__versa_string & insert (size_type __pos, const _CharT *__s)`
- `__versa_string & insert (size_type __pos, const _CharT *__s, size_type __n)`
- `__versa_string & insert (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)`

- [\\_\\_versa\\_string](#) & [insert](#) (size\_type \_\_pos1, const [\\_\\_versa\\_string](#) &\_\_str)
- void [insert](#) (iterator \_\_p, [std::initializer\\_list](#)< \_CharT > \_\_l)
- template<class \_InputIterator >  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- size\_type [length](#) () const
- size\_type [max\\_size](#) () const
- [\\_\\_versa\\_string](#) & [operator+=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [\\_\\_versa\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [\\_\\_versa\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & [operator+=](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [\\_\\_versa\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & [operator=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [\\_\\_versa\\_string](#) & [operator=](#) ([\\_\\_versa\\_string](#) &&\_\_str)
- [\\_\\_versa\\_string](#) & [operator=](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- reference [operator\[\]](#) (size\_type \_\_pos)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const
- void [push\\_back](#) (\_CharT \_\_c)
- [const\\_reverse\\_iterator](#) [rbegin](#) () const
- [reverse\\_iterator](#) [rbegin](#) ()
- [const\\_reverse\\_iterator](#) [rend](#) () const
- [reverse\\_iterator](#) [rend](#) ()
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [std::initializer\\_list](#)< \_CharT > \_\_l)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- template<class \_InputIterator >  
[\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_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 \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- `__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 \_\_n, const \_\_versa\_string & \_\_str)
- 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=npos) const
- 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
- 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`

### 5.22.1 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 52 of file `vstring.h`.

## 5.22.2 Constructor & Destructor Documentation

**5.22.2.1** `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 130 of file `vstring.h`.

**5.22.2.2** `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 137 of file `vstring.h`.

**5.22.2.3** `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 145 of file `vstring.h`.

**5.22.2.4** `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 157 of file `vstring.h`.

```
5.22.2.5 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 165 of file `vstring.h`.

```
5.22.2.6 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 176 of file `vstring.h`.

```
5.22.2.7 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 191 of file `vstring.h`.

```
5.22.2.8 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 208 of file `vstring.h`.

```
5.22.2.9 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 217 of file `vstring.h`.

**5.22.2.10** `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 227 of file `vstring.h`.

**5.22.2.11** `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 237 of file `vstring.h`.

**5.22.2.12** `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 244 of file `vstring.h`.

### 5.22.3 Member Function Documentation

**5.22.3.1** `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 759 of file `vstring.h`.

**5.22.3.2** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (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 745 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()`.

**5.22.3.3** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (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 735 of file `vstring.h`.

**5.22.3.4** `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 718 of file `vstring.h`.

**5.22.3.5** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (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 705 of file `vstring.h`.

```
5.22.3.6 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append (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 693 of file `vstring.h`.

```
5.22.3.7 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 676 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `std::getline()`, and `__gnu_cxx::operator+()`.

**5.22.3.8** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (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 893 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()`.

**5.22.3.9** `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 883 of file vstring.h.

```
5.22.3.10 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::assign (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 869 of file vstring.h.

```
5.22.3.11 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) [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 852 of file vstring.h.



5.22.3.12 `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, 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 836 of file `vstring.h`.

5.22.3.13 `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::assign (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 819 of file `vstring.h`.

**5.22.3.14** `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 798 of file `vstring.h`.

References `std::swap()`.

**5.22.3.15** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (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 782 of file `vstring.h`.

**5.22.3.16** `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 586 of file `vstring.h`.

```
5.22.3.17 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 567 of file `vstring.h`.

```
5.22.3.18 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 624 of file `vstring.h`.

**5.22.3.19** `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 616 of file `vstring.h`.

**5.22.3.20** `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 321 of file `vstring.h`.

**5.22.3.21** `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 310 of file `vstring.h`.

**5.22.3.22** `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 1486 of file `vstring.h`.

**5.22.3.23** `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 478 of file `vstring.h`.

**5.22.3.24** `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 385 of file `vstring.h`.

**5.22.3.25** `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 393 of file `vstring.h`.

**5.22.3.26** `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 506 of file `vstring.h`.

**5.22.3.27** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> int  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::compare (size_type __pos, size_type __n1, const _CharT * __s,  
size_type __n2) const [inline]`

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 527 of file `vstring.tcc`.

References `std::min()`.

```
5.22.3.28 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 __n1, const _CharT * __s)
const [inline]
```

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.

Definition at line 510 of file `vstring.tcc`.

References `std::min()`.

5.22.3.29 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (const _CharT * __s) const [inline]`

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 494 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()`.

5.22.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2) const [inline]`

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 475 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()`.

**5.22.3.31** `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 __n, const __versa_string<  
_CharT,_Traits,_Alloc,_Base > & __str) const [inline]`

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 458 of file `vstring.tcc`.



## 5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >` Class Template Reference

817

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()`.

**5.22.3.32** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (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 1905 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>=()`.

**5.22.3.33** `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 [inline]`

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 253 of file `vstring.tcc`.

**5.22.3.34** `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 402 of file `vstring.h`.

**5.22.3.35** `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 411 of file `vstring.h`.

**5.22.3.36** `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 1496 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()`.

**5.22.3.37** `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 514 of file `vstring.h`.

**5.22.3.38** `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 340 of file `vstring.h`.

**5.22.3.39** `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 329 of file `vstring.h`.

**5.22.3.40** `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 1125 of file `vstring.h`.

**5.22.3.41** `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 1104 of file `vstring.h`.

**5.22.3.42** `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 1088 of file `vstring.h`.

Referenced by `std::getline()`.

**5.22.3.43** `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(_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 `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 292 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()`.

**5.22.3.44** `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 _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 1546 of file `vstring.h`.

```
5.22.3.45 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 __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 1532 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()`.

```
5.22.3.46 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
 (const _CharT * __s, size_type __pos, size_type __n) const
 [inline]
```

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 268 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()`.

```
5.22.3.47 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 (_CharT __c, size_type __pos = 0) const
[inline]
```

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 403 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.48** `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 1790 of file `vstring.h`.

**5.22.3.49** `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 [inline]`

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 390 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.



**5.22.3.50** `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 __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 1760 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()`.

**5.22.3.51** `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(_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 1683 of file `vstring.h`.

```

5.22.3.52 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 1664 of file `vstring.h`.

```

5.22.3.53 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 [inline]

```

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 351 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()`.

5.22.3.54 `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(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 1635 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()`.

5.22.3.55 `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 [inline]`

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 437 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.56 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 _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 1852 of file `vstring.h`.

```
5.22.3.57 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 [inline]
```

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.

## 5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >` Class Template Reference

829

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 415 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.58** `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 __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 1821 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()`.

**5.22.3.59** `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 1746 of file `vstring.h`.

```
5.22.3.60 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 _CharT * __s, size_type __pos = npo) 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 1727 of file `vstring.h`.

```
5.22.3.61 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 [inline]
```

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 368 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.22.3.62** `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(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 1698 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()`.

**5.22.3.63** `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 608 of file `vstring.h`.

**5.22.3.64** `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 600 of file `vstring.h`.

**5.22.3.65** `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 1503 of file `vstring.h`.

**5.22.3.66** `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 1063 of file `vstring.h`.

```
5.22.3.67 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 1045 of file `vstring.h`.

```
5.22.3.68 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 __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 1021 of file `vstring.h`.

**5.22.3.69** `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 1002 of file `vstring.h`.

5.22.3.70 `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 979 of file `vstring.h`.

5.22.3.71 `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 956 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.22.3.72** `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 939 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()`.

**5.22.3.73** `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 928 of file `vstring.h`.

**5.22.3.74** `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 911 of file `vstring.h`.

**5.22.3.75** `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 426 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

**5.22.3.76** `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 431 of file `vstring.h`.

Referenced by `std::getline()`.

**5.22.3.77** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(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 666 of file `vstring.h`.

**5.22.3.78** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(_CharT __c) [inline]`

Append a `character`.

**Parameters:**

`__c` The `character` to append.

**Returns:**

Reference to this string.

Definition at line 653 of file `vstring.h`.

```
5.22.3.79 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator+=(const _CharT * __s) [inline]
```

Append a C string.

**Parameters:**

`__s` The C string to append.

**Returns:**

Reference to this string.

Definition at line 644 of file `vstring.h`.

```
5.22.3.80 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator+=(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 635 of file `vstring.h`.

```
5.22.3.81 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator=(_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 298 of file `vstring.h`.

```
5.22.3.82 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (const _CharT * __s) [inline]
```

Copy contents of `__s` into this string.

**Parameters:**

`__s` Source null-terminated string.

Definition at line 287 of file `vstring.h`.

```
5.22.3.83 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (std::initializer_list< _CharT > __l)
[inline]
```

Set value to string constructed from initializer list.

**Parameters:**

`__l` `std::initializer_list`.

Definition at line 275 of file `vstring.h`.

```
5.22.3.84 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (__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 263 of file `vstring.h`.

References `std::swap()`.

```
5.22.3.85 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 251 of file `vstring.h`.

```
5.22.3.86 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 546 of file `vstring.h`.

```
5.22.3.87 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 529 of file `vstring.h`.

```
5.22.3.88 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 767 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

```
5.22.3.89 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 358 of file `vstring.h`.

```
5.22.3.90 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 349 of file `vstring.h`.

**5.22.3.91** `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 376 of file `vstring.h`.

**5.22.3.92** `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 367 of file `vstring.h`.

**5.22.3.93** `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 1422 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()`.

**5.22.3.94** `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 1354 of file `vstring.h`.

**5.22.3.95** `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace(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 1331 of file `vstring.h`.

**5.22.3.96** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (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 1310 of file `vstring.h`.

**5.22.3.97** `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, 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 1289 of file `vstring.h`.

**5.22.3.98** `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 __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 1271 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()`.

**5.22.3.99** `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 1253 of file `vstring.h`.

**5.22.3.100** `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) [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 first `__n` 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 1229 of file `vstring.h`.

**5.22.3.101** `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, 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 1204 of file `vstring.h`.

**5.22.3.102** `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 1176 of file `vstring.h`.

```

5.22.3.103 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::replace (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 1153 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\(\)](#).

```

5.22.3.104 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 499 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

**5.22.3.105** `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) [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 458 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()`.

**5.22.3.106** `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, _CharT __c) [inline]`

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 49 of file `vstring.tcc`.

```
5.22.3.107 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 (_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`.

Definition at line 333 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.108 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 1605 of file `vstring.h`.

```
5.22.3.109 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
[inline]
```

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 311 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()`.

```
5.22.3.110 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 1576 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 >::rfind()`.

**5.22.3.111** `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 464 of file `vstring.h`.

**5.22.3.112** `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 420 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__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()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

5.22.3.113 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __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 1884 of file `vstring.h`.

5.22.3.114 `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 1475 of file `vstring.h`.

Referenced by `__gnu_cxx::swap()`.

## 5.22.4 Member Data Documentation

**5.22.4.1** `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 [inline, static]`

Value returned by various member functions when they fail.

Definition at line 77 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](#)



## 5.23 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

### Public Types

- `typedef _ToType::element_type * type`

#### 5.23.1 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 45 of file `cast.h`.

The documentation for this struct was generated from the following file:

- `cast.h`

## 5.24 `__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`

### 5.24.1 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 65 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.25 `__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

- `template<typename _Up>`  
`_ExtPtr_allocator` (const `_ExtPtr_allocator<_Up>` &\_\_rarg) throw ()
- `_ExtPtr_allocator` (const `_ExtPtr_allocator` &\_\_rarg) throw ()
- `const std::allocator<_Tp> &` `_M_getUnderlyingImp` () const
- `const_pointer address` (const\_reference \_\_x) const
- `pointer address` (reference \_\_x) const
- `pointer allocate` (size\_type \_\_n, void \* \_\_hint=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 ()
- `bool operator!=` (const `_ExtPtr_allocator` &\_\_rarg)
- `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)

## Friends

- `template<typename _Up >`  
`void swap (_ExtPtr_allocator<_Up > &, _ExtPtr_allocator<_Up > &)`

### 5.25.1 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 52 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

## 5.26 `__gnu_cxx::_Invalid_type` Struct Reference

### 5.26.1 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 205 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.27 `__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

- `template<typename _Up > _Pointer_adapter (const _Pointer_adapter< _Up > &__arg)`
- `template<typename _Up > _Pointer_adapter (_Up * __arg)`
- `_Pointer_adapter (const _Pointer_adapter &__arg)`
- `_Pointer_adapter (element_type * __arg=0)`
- `operator __unspecified_bool_type () const`
- `bool operator! () const`
- `reference operator* () const`
- `_Pointer_adapter operator++ (int __unused)`
- `_Pointer_adapter & operator++ ()`
- `_Pointer_adapter & operator+= (unsigned long __offset)`
- `_Pointer_adapter & operator+= (long __offset)`
- `_Pointer_adapter & operator+= (unsigned int __offset)`
- `_Pointer_adapter & operator+= (int __offset)`
- `_Pointer_adapter & operator+= (unsigned short __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 long __offset)`
- `_Pointer_adapter & operator-= (long __offset)`
- `_Pointer_adapter & operator-= (unsigned int __offset)`
- `_Pointer_adapter & operator-= (int __offset)`

- `_Pointer_adapter` & `operator==` (unsigned short `__offset`)
- `_Pointer_adapter` & `operator==` (short `__offset`)
- `element_type * operator->` () const
- `template<typename _Up > _Pointer_adapter` & `operator=` (`_Up * __arg`)
- `template<typename _Up > _Pointer_adapter` & `operator=` (const `_Pointer_adapter<_Up > & __arg`)
- `_Pointer_adapter` & `operator=` (const `_Pointer_adapter` & `__arg`)
- reference `operator[]` (`std::ptrdiff_t __index`) const

## Friends

- `_Pointer_adapter operator+` (unsigned long `__offset`, const `_Pointer_adapter & __rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, unsigned long `__offset`)
- `_Pointer_adapter operator+` (long `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, long `__offset`)
- `_Pointer_adapter operator+` (unsigned int `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, unsigned int `__offset`)
- `_Pointer_adapter operator+` (int `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, int `__offset`)
- `_Pointer_adapter operator+` (unsigned short `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, unsigned short `__offset`)
- `_Pointer_adapter operator+` (short `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & `__lhs`, short `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, unsigned long `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, long `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, unsigned int `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, int `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, unsigned short `__offset`)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` & `__lhs`, short `__offset`)
- `template<typename _Up > std::ptrdiff_t operator-` (`_Up * __lhs`, const `_Pointer_adapter` & `__rhs`)
- `template<typename _Up > std::ptrdiff_t operator-` (const `_Pointer_adapter` & `__lhs`, `_Up * __rhs`)
- `std::ptrdiff_t operator-` (`element_type * __lhs`, const `_Pointer_adapter` & `__rhs`)
- `std::ptrdiff_t operator-` (const `_Pointer_adapter` & `__lhs`, `element_type * __rhs`)

### 5.27.1 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 281 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)



## 5.28 `__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)`

### 5.28.1 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 101 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.29 `__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)`

### 5.29.1 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 153 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.30 `__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)`

#### 5.30.1 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 58 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.31 `__gnu_cxx::_Unqualified_type< _Tp >` Struct Template Reference

### Public Types

- `typedef _Tp type`

#### 5.31.1 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 233 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.32 `__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`:



### 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` &)

#### 5.32.1 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.

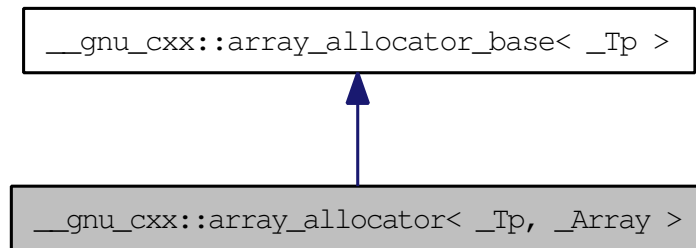
Definition at line 94 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.33 `__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. 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

- `template<typename _Tp1, typename _Array1 >`  
**array\_allocator** (`const array_allocator< _Tp1, _Array1 > &`) `throw ()`
- **array\_allocator** (`const array_allocator &__o`) `throw ()`
- **array\_allocator** (`array_type *__array=NULL`) `throw ()`
- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **address** (`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, size_type`)
- void **destroy** (`pointer __p`)
- `size_type` **max\_size** () `const throw ()`

## 5.33 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference 871

### 5.33.1 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.

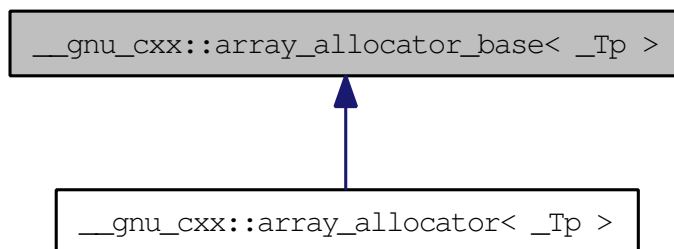
Definition at line 96 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

## 5.34 `__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

- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **address** (`reference __x`) `const`
- `template<typename... _Args>`  
`void` **construct** (`pointer __p`, `_Args &&...__args`)
- `void` **construct** (`pointer __p`, `const _Tp &__val`)
- `void` **deallocate** (`pointer`, `size_type`)
- `void` **destroy** (`pointer __p`)
- `size_type` **max\_size** () `const` `throw ()`

#### 5.34.1 Detailed Description

```
template<typename _Tp> class __gnu_cxx::array_allocator_base< _Tp >
```

Base class.



### **5.34 \_\_gnu\_cxx::array\_allocator\_base<\_Tp> Class Template Reference 873**

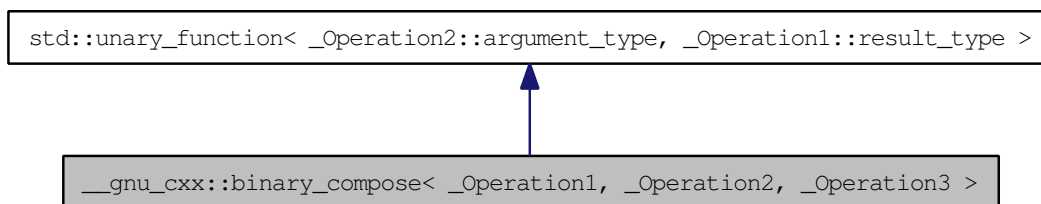
Definition at line 46 of file array\_allocator.h.

The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

### 5.35 `__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**

#### 5.35.1 Detailed Description

`template<class _Operation1, class _Operation2, class _Operation3> class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`

An [SGI extension](#) .

Definition at line 150 of file `ext/functional`.

## 5.35.2 Member Typedef Documentation

**5.35.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function<_Arg, _Result>::argument_type  
[inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.35.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function<_Arg, _Result>::result_type [inherited]`

`result_type` is the return type

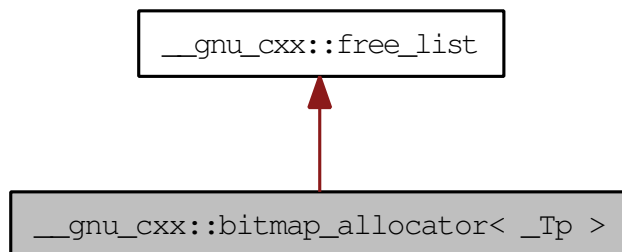
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

### 5.36 `__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

- `template<typename _Tp1 >`  
**bitmap\_allocator** (`const bitmap_allocator<_Tp1 > &`) `throw ()`
- **bitmap\_allocator** (`const bitmap_allocator &`)
- `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 ()

## 5.36.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::bitmap_allocator<_Tp>`

Bitmap Allocator, primary template.

Definition at line 687 of file `bitmap_allocator.h`.

## 5.36.2 Member Function Documentation

**5.36.2.1** `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 821 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan_forward()`.

**5.36.2.2** `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 911 of file `bitmap_allocator.h`.

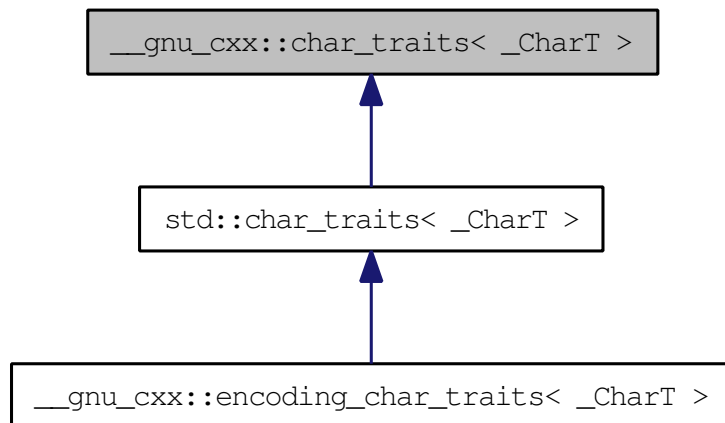
References `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `std::__rotate()`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.37 `__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 `char_type * assign` (`char_type * __s`, `std::size_t __n`, `char_type __a`)
- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- 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 `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`, `std::size_t __n`, `const char_type &__a`)

- static `std::size_t` **length** (`const char_type *__s`)
- static `bool` **It** (`const char_type &__c1, const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1, const char_type *__s2, std::size_t __n`)
- static `int_type` **not\_eof** (`const int_type &__c`)
- static `char_type` **to\_char\_type** (`const int_type &__c`)
- static `int_type` **to\_int\_type** (`const char_type &__c`)

### 5.37.1 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 90 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)



## 5.38 `__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`

#### 5.38.1 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.

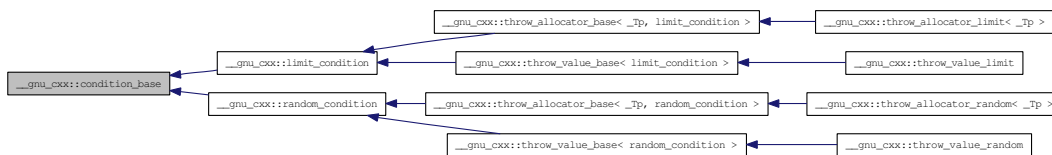
Definition at line 45 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.39 `__gnu_cxx::condition_base` Struct Reference

Base struct for condition policy. Inheritance diagram for `__gnu_cxx::condition_base`:



### 5.39.1 Detailed Description

Base struct for condition policy. Requires a public member function with the signature `void throw_conditionally()`

Definition at line 253 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.40 `__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**

#### 5.40.1 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 316 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.41 `__gnu_cxx::constant_unary_fun< _Result, _Argument >` Struct Template Reference

An [SGI extension](#) .

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**

#### 5.41.1 Detailed Description

```
template<class _Result, class _Argument = _Result> struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
```

An [SGI extension](#) .

Definition at line 308 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.42 `__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`

#### 5.42.1 Detailed Description

`template<class _Result> struct __gnu_cxx::constant_void_fun<_Result>`

An [SGI extension](#).

Definition at line 299 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.43 `__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, const void \*\_\_hint)
- pointer **allocate** (size\_type \_\_n)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)

#### 5.43.1 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 61 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 5.44 `__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`)
- void `stoss` ()
- `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 ()
- 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 `xsggetn` (char\_type \*\_\_s, streamsize \_\_n)
- virtual streamsize `xspun` (const char\_type \*\_\_s, streamsize \_\_n)

## 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>` &, `basic_string<_CharT2, _Traits2, _Alloc>` &)
  - `basic_istream<_CharT2, _Traits2>` & `operator>>` (`basic_istream<_CharT2, _Traits2>` &, `_CharT2 *`)
  - class `ostreambuf_iterator<char_type, traits_type>`
- 
- locale `getloc` () const
  - locale `pubimbue` (const locale &\_\_loc)
  - `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`)
  - `__streambuf_type *` `pubsetbuf` (`char_type *` \_\_s, `streamsize` \_\_n)

- [int pubsync \(\)](#)
- [locale \\_M\\_buf\\_locale](#)
- [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](#)
- [char\\_type \\* eback \(\) const](#)
- [char\\_type \\* egptr \(\) const](#)
- [char\\_type \\* epptr \(\) const](#)
- [char\\_type \\* gptr \(\) const](#)
- [char\\_type \\* pbase \(\) const](#)
- [char\\_type \\* pptr \(\) const](#)

### 5.44.1 Detailed Description

`template<typename _CharT> class __gnu_cxx::enc_filebuf< _CharT >`

class [enc\\_filebuf](#).

Definition at line 40 of file `enc_filebuf.h`.

### 5.44.2 Member Typedef Documentation

**5.44.2.1** `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 77 of file `fstream`.

**5.44.2.2** `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 71 of file `fstream`.

**5.44.2.3** `typedef traits_type::int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type [inherited]`

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 73 of file `fstream`.

**5.44.2.4** `typedef traits_type::off_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::off_type [inherited]`

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 75 of file `fstream`.

**5.44.2.5** `template<typename _CharT> typedef traits_type::pos_type __gnu_cxx::enc_filebuf<_CharT>::pos_type`

This is a non-standard type.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 46 of file `enc_filebuf.h`.

**5.44.2.6** `template<typename _CharT> typedef encoding_char_traits<_CharT> __gnu_cxx::enc_filebuf<_CharT>::traits_type`

This is a non-standard type.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 44 of file `enc_filebuf.h`.

### 5.44.3 Member Function Documentation

**5.44.3.1** `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 172 of file `fstream`.

**5.44.3.2** `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 189 of file `fstream`.

**5.44.3.3** `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 387 of file `fstream`.

**5.44.3.4** `__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.

**5.44.3.5** `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 460 of file `streambuf`.

**5.44.3.6** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr() const [inline, protected, 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 466 of file `streambuf`.

**5.44.3.7** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eptr() const [inline, protected, 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 513 of file `streambuf`.

**5.44.3.8** `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 476 of file streambuf.

**5.44.3.9** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::getloc() const [inline, inherited]`

Locale access.

**Returns:**

The current locale in effect.

If `pubimbue(locale)` has been called, then the most recent `locale` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

**5.44.3.10** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr() const [inline, protected, inherited]`

Locale access.

**Returns:**

The current locale in effect.

If `pubimbue(locale)` has been called, then the most recent `locale` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 463 of file streambuf.

**5.44.3.11** `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>>`.

**5.44.3.12** `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 262 of file `streambuf`.

**5.44.3.13** `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 222 of file `fstream`.

**5.44.3.14** `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open(const std::string &_s, ios_base::openmode __mode)` [`inline`, `inherited`]

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 275 of file `fstream`.

**5.44.3.15** `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::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 | +-----+

**5.44.3.16** `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>>`.

**5.44.3.17** `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>>`.

**5.44.3.18** `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 507 of file `streambuf`.

**5.44.3.19** `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 523 of file `streambuf`.

**5.44.3.20** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr() const [inline, protected, 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 510 of file `streambuf`.

**5.44.3.21** `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 205 of file `streambuf`.

**5.44.3.22** `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]`

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 239 of file `streambuf`.

**5.44.3.23** `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]`

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 244 of file `streambuf`.

**5.44.3.24** `__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 235 of file `streambuf`.

#### 5.44.3.25 `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync() [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 249 of file `streambuf`.

#### 5.44.3.26 `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 294 of file `streambuf`.

#### 5.44.3.27 `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>>`.

**5.44.3.28** `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>>](#).

**5.44.3.29** `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>>](#).

**5.44.3.30** `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 487 of file `streambuf`.

**5.44.3.31** `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 533 of file `streambuf`.

**5.44.3.32** `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 316 of file `streambuf`.

**5.44.3.33** `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 335 of file `streambuf`.

**5.44.3.34** `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>>`.

**5.44.3.35** `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 276 of file `streambuf`.

**5.44.3.36** `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 350 of file `streambuf`.

**5.44.3.37** `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 402 of file `streambuf`.

**5.44.3.38** `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 `xspu(n,s)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.44.3.39** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::stoss() [inline, inherited]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.44.3.40** `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 `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

**5.44.3.41** `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>>`.

**5.44.3.42** `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 678 of file `streambuf`.

**5.44.3.43** `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>>`.

**5.44.3.44** `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>>`.

**5.44.3.45** `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 > >`.

#### 5.44.4 Member Data Documentation

##### 5.44.4.1 `char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf` [protected, inherited]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

##### 5.44.4.2 `locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

##### 5.44.4.3 `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 116 of file `fstream`.

##### 5.44.4.4 `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 151 of file `fstream`.

##### 5.44.4.5 `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 156 of file `fstream`.

**5.44.4.6** `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 163 of file `fstream`.

**5.44.4.7** `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 180 of file `streambuf`.

**5.44.4.8** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` [protected, 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 181 of file `streambuf`.

**5.44.4.9** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_end` [protected, 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 182 of file `streambuf`.

**5.44.4.10** `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 94 of file `fstream`.

**5.44.4.11** `char_type*` `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg` [`protected`, `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 183 of file `streambuf`.

**5.44.4.12** `char_type*` `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur` [`protected`, `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 184 of file `streambuf`.

**5.44.4.13** `char_type*` `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end` [`protected`, `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 185 of file `streambuf`.

**5.44.4.14** `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 137 of file `fstream`.

**5.44.4.15** `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 138 of file `fstream`.

**5.44.4.16** `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 139 of file `fstream`.

**5.44.4.17** `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_init` [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`.

**5.44.4.18** `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 128 of file `fstream`.

The documentation for this class was generated from the following file:

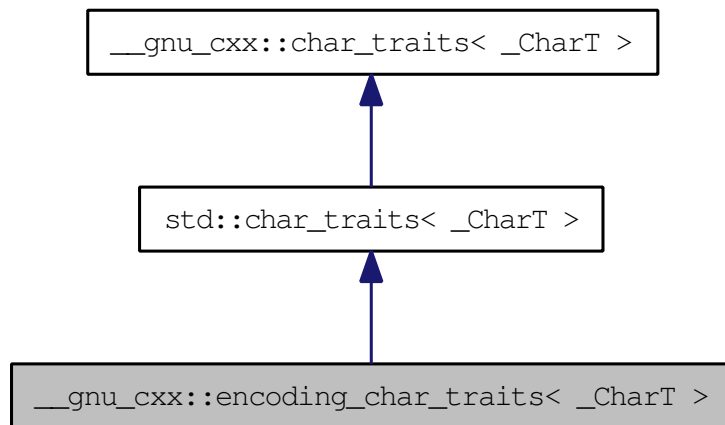
- [enc\\_filebuf.h](#)



## 5.45 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

### 5.45 `__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 `char_type * assign (char_type * __s, std::size_t __n, char_type __a)`
- static void `assign (char_type & __c1, const char_type & __c2)`
- 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 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, std::size_t __n, const char_type & __a)`

- static std::size\_t **length** (const char\_type \*\_\_s)
- static bool **It** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static char\_type **to\_char\_type** (const int\_type &\_\_c)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.45.1 Detailed Description

```
template<typename _CharT> struct __gnu_cxx::encoding_char_traits< _-
CharT >
```

[encoding\\_char\\_traits](#)

Definition at line 209 of file `codecvt_specializations.h`.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.46 `__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 `encoding_state` &\_\_obj)
- `encoding_state` (const char \*\_\_int, const char \*\_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- 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`

### 5.46.1 Detailed Description

Extension to use iconv for dealing with [character](#) encodings.

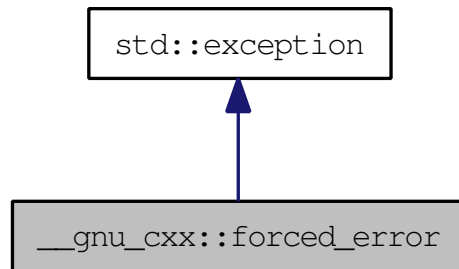
Definition at line 49 of file `codecv_t_specializations.h`.

The documentation for this class was generated from the following file:

- [codecv\\_t\\_specializations.h](#)

## 5.47 `__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 ()

#### 5.47.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 73 of file `throw_allocator.h`.

#### 5.47.2 Member Function Documentation

##### 5.47.2.1 virtual const char\* `std::exception::what` () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

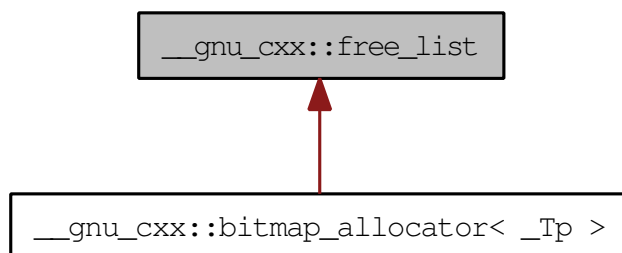
Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::bad_weak_ptr`, and `std::ios_base::failure`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.48 `__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 ()

#### 5.48.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).

Definition at line 521 of file `bitmap_allocator.h`.

#### 5.48.2 Member Function Documentation

##### 5.48.2.1 void `__gnu_cxx::free_list::_M_clear` ()

This function just clears the internal Free List, and gives back all the memory to the OS.

**5.48.2.2** `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.

**5.48.2.3** `void __gnu_cxx::free_list::_M_insert (size_t * __addr) throw () [inline]`

This function returns the block of memory to the internal free list.

**Parameters:**

`__addr` The pointer to the memory block that was given by a call to the `_M_get` function.

Definition at line 631 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.49 `__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

- 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()`)
- 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`)
- template<class `_InputIterator` >  
**hash\_map** (`_InputIterator __f`, `_InputIterator __l`)
- **hash\_map** (`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_iterator` **begin** () const
- `iterator` **begin** ()
- `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`
- `const_iterator end () const`
- `iterator end ()`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `pair< iterator, iterator > equal_range (const key_type &__key)`
- `void erase (iterator __f, iterator __l)`
- `void erase (iterator __it)`
- `size_type erase (const key_type &__key)`
- `const_iterator find (const key_type &__key) const`
- `iterator find (const key_type &__key)`
- `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 _Al > bool operator== (const hash_map<_K1, _T1, _HF, _EqK, _Al > &, const hash_map<_K1, _T1, _HF, _EqK, _Al > &)`

### 5.49.1 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 78 of file hash\_map.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.50 `__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

- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n`, `const hasher &__hf`, `const key_equal &__eql`, `const allocator_type &__a=allocator_type()`)
- `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`)
- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f`, `_InputIterator __l`)
- `hash_multimap` (`size_type __n`, `const hasher &__hf`, `const key_equal &__eql`, `const allocator_type &__a=allocator_type()`)
- `hash_multimap` (`size_type __n`, `const hasher &__hf`)
- `hash_multimap` (`size_type __n`)
- `const_iterator begin` () const
- `iterator begin` ()
- `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
- const\_iterator **end** () const
- iterator **end** ()
- pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- size\_type **erase** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- iterator **find** (const key\_type &\_\_key)
- allocator\_type **get\_allocator** () const
- hasher **hash\_func** () 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 > &)

### 5.50.1 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.

**5.50 `__gnu_cxx::hash_multimap`< `_Key`, `_Tp`, `_HashFn`, `_EqualKey`, `_Alloc` >**  
**Class Template Reference** **925**

---

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 291 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.51 `__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

- 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()`)
- 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`)
- template<class `_InputIterator` >  
**hash\_multiset** (`_InputIterator __f`, `_InputIterator __l`)
- **hash\_multiset** (`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`)
- **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_func ()` 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 > &)`

### 5.51.1 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 280 of file `hash_set`.

The documentation for this class was generated from the following file:

- `hash_set`

## 5.52 `__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

- `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())`
- `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)`
- `template<class _InputIterator >`  
`hash_set (_InputIterator __f, _InputIterator __l)`
- `hash_set (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)`
- `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 > &)

### 5.52.1 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

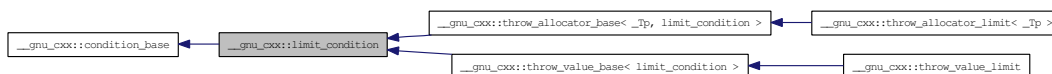
Definition at line 79 of file `hash_set`.

The documentation for this class was generated from the following file:

- `hash_set`

## 5.53 `__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 ()`

#### 5.53.1 Detailed Description

Base class for incremental control and throw.

Definition at line 262 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.54 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### 5.54.1 Detailed Description

Always enter the condition.

Definition at line 286 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.55 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Enter the nth condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### Public Member Functions

- `limit_adjustor` (const size\_t \_\_l)

#### 5.55.1 Detailed Description

Enter the nth condition.

Definition at line 292 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.56 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### 5.56.1 Detailed Description

Never enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.57 `__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

- template<typename \_Tp1 >  
**malloc\_allocator** (const [malloc\\_allocator](#)< \_Tp1 > &) throw ()
- **malloc\_allocator** (const [malloc\\_allocator](#) &) throw ()
- const\_pointer **address** (const\_reference \_\_x) const
- pointer **address** (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)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

#### 5.57.1 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 52 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc\\_allocator.h](#)

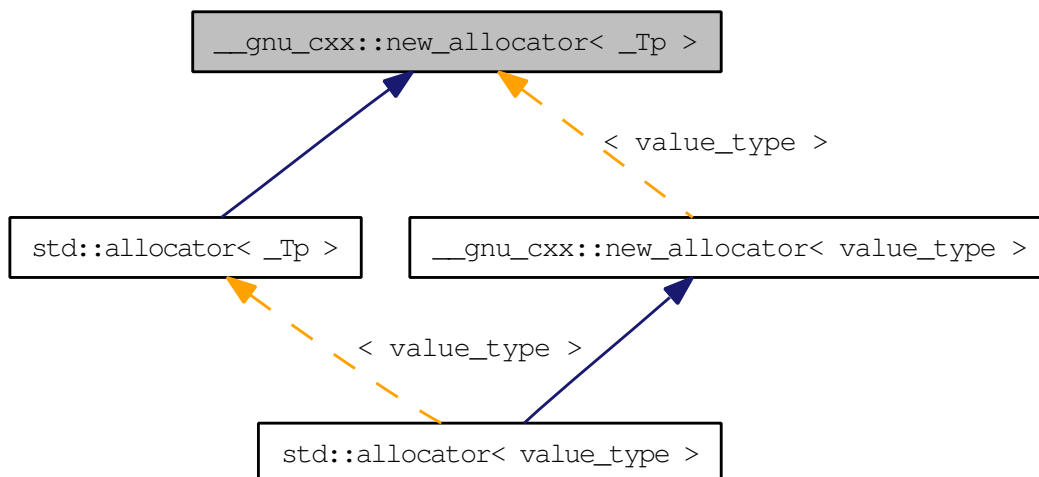
## 5.58 `__gnu_cxx::new_allocator< _Tp >` Class Template Reference

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

- `template<typename _Tp1 >`  
**new\_allocator** (`const new_allocator< _Tp1 > &`) `throw ()`



- `new_allocator` (const `new_allocator` &) throw ()
- const\_pointer `address` (const\_reference \_\_x) const
- pointer `address` (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)
- void `destroy` (pointer \_\_p)
- size\_type `max_size` () const throw ()

### 5.58.1 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 51 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new\\_allocator.h](#)

## 5.59 `__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

#### 5.59.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 233 of file `ext/functional`.

#### 5.59.2 Member Typedef Documentation

**5.59.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`  
`>::first_argument_type` [`inherited`]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.59.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`  
`>::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.59.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.60 `__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

#### 5.60.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 237 of file `ext/functional`.

#### 5.60.2 Member Typedef Documentation

**5.60.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`  
`>::first_argument_type` [`inherited`]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.60.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`  
`>::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.60.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

the type of the second argument

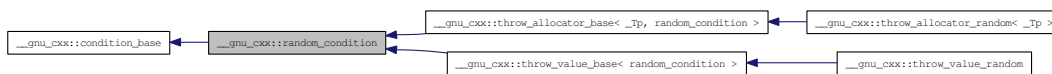
Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.61 `__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** ()

#### 5.61.1 Detailed Description

Base class for random probability control and throw.

Definition at line 334 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.62 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.62.1 Detailed Description

Always enter the condition.

Definition at line 367 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.63 `__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)

#### 5.63.1 Detailed Description

Group condition.

Definition at line 352 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)



## 5.64 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.64.1 Detailed Description

Never enter the condition.

Definition at line 361 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.65 `__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<class _II >  
void _M_insert_equal (_II __first, _II __last)`
- `template<typename _InputIterator >  
void _M_insert_equal (_InputIterator __first, _InputIterator __last)`
- `iterator _M_insert_equal (const value_type &__x)`
- `iterator _M_insert_equal (const_iterator __position, const value_type &__x)`
- `template<class _II >  
void _M_insert_unique (_II __first, _II __last)`

- `template<typename _InputIterator >`  
`void _M_insert_unique (_InputIterator __first, _InputIterator __last)`
- `pair< iterator, bool > _M_insert_unique (const value_type &__x)`
- `iterator _M_insert_unique_ (const_iterator __position, const value_type &__x)`
  
- `const_iterator begin () const`
- `iterator begin ()`
- `void clear ()`
- `size_type count (const key_type &__k) const`
- `bool empty () const`
- `const_iterator end () const`
- `iterator end ()`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
  
- `pair< iterator, iterator > equal_range (const key_type &__k)`
- `void erase (const key_type *__first, const key_type *__last)`
- `const_iterator erase (const_iterator __first, const_iterator __last)`
- `iterator erase (iterator __first, iterator __last)`
- `size_type erase (const key_type &__x)`
- `const_iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
- `const_iterator find (const key_type &__k) const`
- `iterator find (const key_type &__k)`
- `allocator_type get_allocator () const`
- `_Compare key_comp () const`
- `const_iterator lower_bound (const key_type &__k) const`
- `iterator lower_bound (const key_type &__k)`
- `size_type max_size () const`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `size_type size () const`
- `void swap (_Rb_tree &__t)`
- `const_iterator upper_bound (const key_type &__k) const`
- `iterator upper_bound (const key_type &__k)`

## Protected Types

- `typedef _Rb_tree_node_base * _Base_ptr`
- `typedef const _Rb_tree_node_base * _Const_Base_ptr`

## Protected Member Functions

- `_Const_Link_type _M_begin () const`
- `_Link_type _M_begin ()`
- `_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 ()`
- `_Const_Base_ptr _M_leftmost () const`
- `_Base_ptr & _M_leftmost ()`
- `void _M_put_node (_Link_type __p)`
- `_Const_Base_ptr _M_rightmost () const`
- `_Base_ptr & _M_rightmost ()`
- `_Const_Base_ptr _M_root () const`
- `_Base_ptr & _M_root ()`

## Static Protected Member Functions

- `static const _Key & _S_key (_Const_Base_ptr __x)`
- `static const _Key & _S_key (_Const_Link_type __x)`
- `static _Const_Link_type _S_left (_Const_Base_ptr __x)`
- `static _Link_type _S_left (_Base_ptr __x)`
- `static _Const_Base_ptr _S_maximum (_Const_Base_ptr __x)`
- `static _Base_ptr _S_maximum (_Base_ptr __x)`
- `static _Const_Base_ptr _S_minimum (_Const_Base_ptr __x)`
- `static _Base_ptr _S_minimum (_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_Base_ptr __x)`
- `static const_reference _S_value (_Const_Link_type __x)`

## Protected Attributes

- `_Rb_tree_impl< _Compare > _M_impl`

### 5.65.1 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 78 of file `rb_tree`.

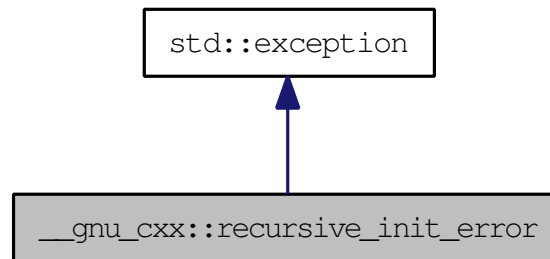
The documentation for this struct was generated from the following file:

- [rb\\_tree](#)

## 5.66 `__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 ()

#### 5.66.1 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 620 of file `cxxabi.h`.

#### 5.66.2 Member Function Documentation

##### 5.66.2.1 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::bad_weak_ptr`, and `std::ios_base::failure`.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 5.67 `__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 [rope](#) &\_\_x, 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 allocator\_type &\_\_a=allocator\_type())
- `rope` (size\_t \_\_n, \_CharT \_\_c, const allocator\_type &\_\_a=allocator\_type())
- `rope` (\_CharT \_\_c, const allocator\_type &\_\_a=allocator\_type())
- `rope` (const iterator &\_\_s, const iterator &\_\_e, const allocator\_type &\_\_a=allocator\_type())
- `rope` (const const\_iterator &\_\_s, const const\_iterator &\_\_e, const allocator\_type &\_\_a=allocator\_type())
- `rope` (const \_CharT \*\_\_s, const \_CharT \*\_\_e, const allocator\_type &\_\_a=allocator\_type())



- `rope` (const `_CharT *__s`, `size_t __len`, const `allocator_type &__a=allocator_type()`)
- `rope` (const `_CharT *__s`, const `allocator_type &__a=allocator_type()`)
- const `allocator_type &_M_get_allocator ()` const
- `allocator_type &_M_get_allocator ()`
- `rope & append` (`size_t __n`, `_CharT __c`)
- `rope & append` (const `rope &__y`)
- `rope & append ()`
- `rope & append` (`_CharT __c`)
- `rope & append` (const `iterator __s`, const `iterator __e`)
- `rope & append` (const `_CharT *__s`, const `_CharT *__e`)
- `rope & append` (const `_CharT *__c_string`)
- `rope & append` (const `_CharT *__iter`, `size_t __n`)
- 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 `iterator begin ()` const
- 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
- `size_type copy` (`size_type __pos`, `size_type __n`, `_CharT *__buffer`) const
- void `copy` (`_CharT *__buffer`) const
- void `delete_c_str ()`
- void `dump ()`
- bool `empty ()` const
- const `iterator end ()`
- const `iterator end ()` const
- `iterator erase` (const `iterator &__p`)
- `iterator erase` (const `iterator &__p`, const `iterator &__q`)
- void `erase` (`size_t __p`)
- void `erase` (`size_t __p`, `size_t __n`)
- `size_type find` (const `_CharT *__s`, `size_type __pos=0`) const
- `size_type find` (`_CharT __c`, `size_type __pos=0`) const
- `_CharT front ()` const
- `allocator_type get_allocator ()` const
- `iterator insert` (const `iterator &__p`, const `iterator &__i`, const `iterator &__j`)

- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, size\_t \_\_n, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, const rope &\_\_r)
- void **insert** (size\_t \_\_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, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_t \_\_p)
- void **insert** (size\_t \_\_p, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **insert** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const rope &\_\_r)
- 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** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- void **replace** (const iterator &\_\_p, const\_iterator \_\_i, const\_iterator \_\_j)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const rope &\_\_r)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (size\_t \_\_p, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, \_CharT \_\_c)
- 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, size\_t \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- 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, size\_t \_\_n, const [rope](#) &\_\_r)
- 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** (iterator \_\_start, iterator \_\_end) const
- [rope](#) **substr** (size\_t \_\_start, size\_t \_\_len=1) 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`, `_CharT2 __right`)
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2>` **operator+** (const `rope<_CharT2, _Alloc2>` &`__left`, const `_CharT2 *``__right`)
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2>` **operator+** (const `rope<_CharT2, _Alloc2>` &`__left`, const `rope<_CharT2, _Alloc2>` &`__right`)

### 5.67.1 Detailed Description

```
template<class _CharT, class _Alloc> class __gnu_cxx::rope< _CharT, _Alloc >
```

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 1508 of file rope.

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

## 5.68 `__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

- `const _Pair::first_type & operator() (const _Pair &__x) const`
- `_Pair::first_type & operator() (_Pair &__x) const`

#### 5.68.1 Detailed Description

```
template<class _Pair> struct __gnu_cxx::select1st<_Pair >
```

An [SGI extension](#) .

Definition at line 199 of file `ext/functional`.

#### 5.68.2 Member Typedef Documentation

**5.68.2.1** `typedef _Pair std::unary_function<_Pair, _Pair::first_type >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.68.2.2** `typedef _Pair::first_type std::unary_function<_Pair, _Pair::first_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.69 `__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

- `const _Pair::second_type & operator() (const _Pair &__x) const`
- `_Pair::second_type & operator() (_Pair &__x) const`

#### 5.69.1 Detailed Description

```
template<class _Pair> struct __gnu_cxx::select2nd<_Pair>
```

An [SGI extension](#) .

Definition at line 203 of file `ext/functional`.

#### 5.69.2 Member Typedef Documentation

**5.69.2.1** `typedef _Pair std::unary_function<_Pair, _Pair::second_type>::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.69.2.2** `typedef _Pair::second_type std::unary_function<_Pair, _Pair::second_type>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)



## 5.70 `__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 **slist** &\_\_x)
- template<class `_InputIterator` >  
**slist** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a=`allocator_type`())
- **slist** (`size_type` \_\_n)
- **slist** (`size_type` \_\_n, const `value_type` &\_\_x, const `allocator_type` &\_\_a=`allocator_type`())
- **slist** (const `allocator_type` &\_\_a=`allocator_type`())
- template<class `_InputIterator` >  
void **\_M\_assign\_dispatch** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_false_type`)
- template<class `_Integer` >  
void **\_M\_assign\_dispatch** (`_Integer` \_\_n, `_Integer` \_\_val, `_true_type`)
- void **\_M\_fill\_assign** (`size_type` \_\_n, const `_Tp` &\_\_val)
- template<class `_InputIterator` >  
void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (`size_type` \_\_n, const `_Tp` &\_\_val)
- const\_iterator **before\_begin** () const
- iterator **before\_begin** ()
- const\_iterator **begin** () const
- iterator **begin** ()
- void **clear** ()

- bool **empty** () const
- const\_iterator **end** () const
- iterator **end** ()
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_pos)
- iterator **erase\_after** (iterator \_\_before\_first, iterator \_\_last)
- iterator **erase\_after** (iterator \_\_pos)
- const\_reference **front** () const
- reference **front** ()
- allocator\_type **get\_allocator** () const
- template<class \_InIterator >  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- iterator **insert** (iterator \_\_pos)
- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- iterator **insert\_after** (iterator \_\_pos)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- size\_type **max\_size** () const
- template<class \_StrictWeakOrdering >  
void **merge** (slist &, \_StrictWeakOrdering)
- void **merge** (slist &\_\_x)
- slist & **operator=** (const slist &\_\_x)
- void **pop\_front** ()
- const\_iterator **previous** (const\_iterator \_\_pos) const
- iterator **previous** (const\_iterator \_\_pos)
- 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)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **reverse** ()
- size\_type **size** () const
- template<class \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **sort** ()
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_pos, slist &\_\_x)

- 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 *`, `_Slist_node_base *`)
- `_Slist_node_base * _M_erase_after` (`_Slist_node_base *`, `__pos`)
- `_Slist_node<_Tp> * _M_get_node` ()
- void `_M_put_node` (`_Slist_node<_Tp> *`, `__p`)

### Private Attributes

- `_Slist_node_base` **\_M\_head**

#### 5.70.1 Detailed Description

`template<class _Tp, class _Alloc = allocator<_Tp>> class __gnu_cxx::slist<_Tp, _Alloc>`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 291 of file `slist`.

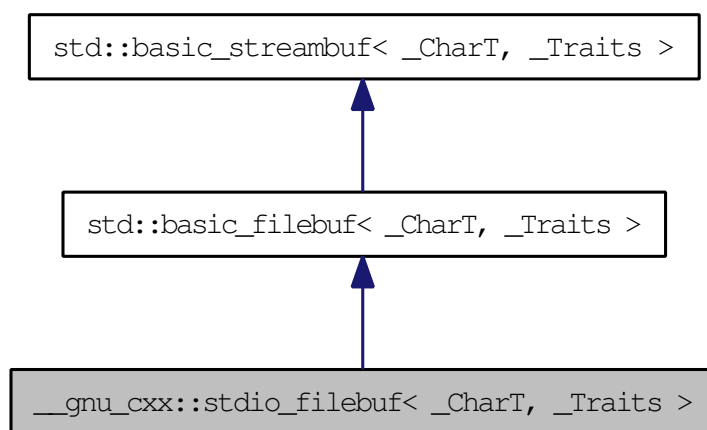
The documentation for this class was generated from the following file:

- [slist](#)

## 5.71 `__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` (`std::_c_file * __f`, `std::ios_base::openmode __mode`, `size_t __size=static_cast< size_t >(BUFSIZ)`)

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 965

- `stdio_filebuf` (int `__fd`, `std::ios_base::openmode` `__mode`, `size_t` `__size=static_cast<size_t>(BUFSIZ)`)
- `stdio_filebuf` ()
- virtual `~stdio_filebuf` ()
- `__filebuf_type * close` ()
- int `fd` ()
- `std::__c_file * file` ()
- `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`)
- void `stoss` ()
- 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 ()
- `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 `xsggetn` (`char_type` \*\_\_s, streamsize \_\_n)
- virtual streamsize `xspun` (const `char_type` \*\_\_s, streamsize \_\_n)

### 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 >`

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 967

- 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, 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 `getloc ()` const
  - locale `pubimbue` (const locale &\_\_loc)
  - `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`)
  - `__streambuf_type * pubsetbuf` (`char_type *__s`, `streamsize __n`)
  - int `pubsync ()`
  - locale `_M_buf_locale`
  - `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`
  - `char_type * eback ()` const
  - `char_type * egptr ()` const
  - `char_type * epptr ()` const
  - `char_type * gptr ()` const
  - `char_type * pbase ()` const
  - `char_type * pptr ()` const
-

### 5.71.1 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 49 of file `stdio_filebuf.h`.

### 5.71.2 Member Typedef Documentation

```
5.71.2.1 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 77 of file `fstream`.

```
5.71.2.2 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 53 of file `stdio_filebuf.h`.

```
5.71.2.3 template<typename _CharT , typename _Traits =
std::char_traits<_CharT>> typedef traits_type::int_type
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::__int_type
```

This is a non-standard type.

Reimplemented from [std::basic\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 55 of file `stdio_filebuf.h`.



## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits >` Class Template Reference 969

**5.71.2.4** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::off_type  
__gnu_cxx::stdio_filebuf<_CharT, _Traits >::off_type`

This is a non-standard type.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 57 of file `stdio_filebuf.h`.

**5.71.2.5** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::pos_type  
__gnu_cxx::stdio_filebuf<_CharT, _Traits >::pos_type`

This is a non-standard type.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 56 of file `stdio_filebuf.h`.

**5.71.2.6** `template<typename _CharT, typename _Traits = std::char_traits<_  
_CharT>> typedef _Traits __gnu_cxx::stdio_filebuf<_CharT, _Traits  
>::traits_type`

This is a non-standard type.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 54 of file `stdio_filebuf.h`.

### **5.71.3 Constructor & Destructor Documentation**

**5.71.3.1** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf<_CharT,  
_Traits >::stdio_filebuf () [inline]`

deferred initialization

Definition at line 64 of file `stdio_filebuf.h`.

```

5.71.3.2 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)) [inline]

```

**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 126 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()`.

```

5.71.3.3 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)) [inline]

```

**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 142 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()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 971

**5.71.3.4** `template<typename _CharT, typename _Traits >`  
`__gnu_cxx::stdio_filebuf<_CharT, _Traits >::~~stdio_filebuf ()`  
`[inline, virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 121 of file `stdio_filebuf.h`.

### 5.71.4 Member Function Documentation

**5.71.4.1** `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 172 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::pbackfail()`.

**5.71.4.2** `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 189 of file `fstream`.

Referenced by `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()`.

**5.71.4.3** `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 387 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 >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### 5.71.4.4 `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT, _Traits >::close () [inline, 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 128 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### 5.71.4.5 `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 460 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 973

**5.71.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::egptr() const` [`inline`,  
`protected`, `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 466 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.71.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::epptr() const` [`inline`,  
`protected`, `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 513 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.71.4.8** `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 107 of file `stdio_filebuf.h`.

**5.71.4.9** `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 117 of file `stdio_filebuf.h`.

**5.71.4.10** `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 476 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf<_CharT, _Traits >::xsgetn()`.

**5.71.4.11** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits >::getloc () const  
[inline, inherited]`

Locale access.

**Returns:**

The current `locale` in effect.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits >` Class Template Reference 975

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 222 of file `streambuf`.

```
5.71.4.12 template<typename _CharT, typename _Traits> char_type*
 std::basic_streambuf<_CharT, _Traits>::gptr () const [inline,
 protected, 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 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf<_CharT, _Traits >::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf<_CharT, _Traits >::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf<_CharT, _Traits >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::xsgetn()`.

```
5.71.4.13 template<typename _CharT, typename _Traits> void
 std::basic_filebuf<_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 from `std::basic_streambuf<_CharT, _Traits >`.

Definition at line 855 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()`.

**5.71.4.14** `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 262 of file `streambuf`.

**5.71.4.15** `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 222 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::close()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `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()`.

**5.71.4.16** `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.

**Parameters:**

*s* The name of the file.



## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 977

*mode* The open mode flags.

### Returns:

`this` on success, `NULL` on failure

Definition at line 275 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::open()`.

**5.71.4.17** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open(const char * __s, ios_base::openmode __mode) [inline, 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 94 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()`.

**5.71.4.18** `template<typename _CharT, typename _Traits > basic_filebuf<_CharT, _Traits >::int_type std::basic_filebuf<_CharT, _Traits >::overflow(int_type = _Traits::eof())` [`inline`, `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 408 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::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 >::sync()`.

**5.71.4.19** `template<typename _CharT, typename _Traits > basic_filebuf<_CharT, _Traits >::int_type std::basic_filebuf<_CharT, _Traits >::pbackfail(int_type = _Traits::eof())` [`inline`, `protected`, `virtual`, `inherited`]

Tries to back up the input sequence.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 979

### 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 356 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::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>::seekoff()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

### 5.71.4.20 `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 507 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_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.71.4.21** `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 523 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()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.71.4.22** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr () const [inline,  
protected, 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 510 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_filebuf< _CharT, _Traits >::sync()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.71.4.23** `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](#).

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 981

### Returns:

The previous [locale](#).

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

```
5.71.4.24 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]
```

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 239 of file `streambuf`.

```
5.71.4.25 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]
```

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 244 of file `streambuf`.

```
5.71.4.26 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. 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 235 of file `streambuf`.

**5.71.4.27** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync () [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 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.71.4.28** `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 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

**5.71.4.29** `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) [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.

## 5.71 `gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 983

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 686 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_filebuf<_CharT, _Traits>::pbackfail()`.

```
5.71.4.30 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) [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 from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 739 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()`.

```
5.71.4.31 template<typename _CharT, typename _Traits> basic_filebuf<
 _CharT, _Traits>::__streambuf_type * std::basic_filebuf<_CharT,
 _Traits>::setbuf(char_type * __s, streamsize __n) [inline,
 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 657 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()`.

**5.71.4.32** `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* == `gptra()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

**5.71.4.33** `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* == `eptra()`

Definition at line 533 of file `streambuf`.



## 5.71 `gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 985

**5.71.4.34** `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 316 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()`.

**5.71.4.35** `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 335 of file `streambuf`.

**5.71.4.36** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_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 from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 178 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()`.

**5.71.4.37** `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 276 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()`.

**5.71.4.38** `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.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 987

### **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 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.71.4.39** `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 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.71.4.40** `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 `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.71.4.41** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits >::stoss() [inline,  
inherited]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.71.4.42** `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 375 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits >::ungetc()`.

**5.71.4.43** `template<typename _CharT, typename _Traits > int  
std::basic_filebuf<_CharT, _Traits >::sync() [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*.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 989

### **Note:**

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 838 of file `fstream.tcc`.

References [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::pbase\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::pptr\(\)](#).

**5.71.4.44** `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 678 of file `streambuf`.

**5.71.4.45** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits  
>::underflow () [inline, 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 204 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`, and `std::min()`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::pbackfail()`.

**5.71.4.46** `template<typename _CharT, typename _Traits > streamsize  
std::basic_filebuf<_CharT, _Traits >::xsgetn(char_type *  
__s, streamsize __n) [inline, 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 >`.

## 5.71 `gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 991

Definition at line 527 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`, and `std::basic_streambuf<char_type, traits_type>::xsgetn()`.

**5.71.4.47** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::xsputn(const char_type *  
_s, streamsize _n)` [`inline`, `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 610 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>::pptr()`, `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()`.

## 5.71.5 Member Data Documentation

**5.71.5.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_buf` [**protected**,  
**inherited**]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`.

**5.71.5.2** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::_M_buf_locale` [**protected**,  
**inherited**]

Current [locale](#) setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`.

**5.71.5.3** `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 116 of file `fstream`.

Referenced by `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()`.

**5.71.5.4** `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 151 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.



## 5.71 `_gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 993

**5.71.5.5** `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 156 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.6** `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 163 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.7** `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 180 of file `streambuf`.

**5.71.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_cur  
[protected, 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 181 of file streambuf.

**5.71.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, 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 182 of file streambuf.

**5.71.5.10** `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 94 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 >::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()`.

**5.71.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, 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 183 of file streambuf.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 995

**5.71.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_cur  
[protected, 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 184 of file `streambuf`.

**5.71.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_end  
[protected, 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 185 of file `streambuf`.

**5.71.5.14** `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 137 of file `fstream`.

**5.71.5.15** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf<_CharT, _Traits>::_M_pback_cur_save  
[protected, inherited]`

Necessary bits for putback buffer management.

**Note:**

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

**5.71.5.16** `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 139 of file fstream.

**5.71.5.17** `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 140 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.71.5.18** `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 128 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`,

## **5.71 \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits > Class Template Reference 997**

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().

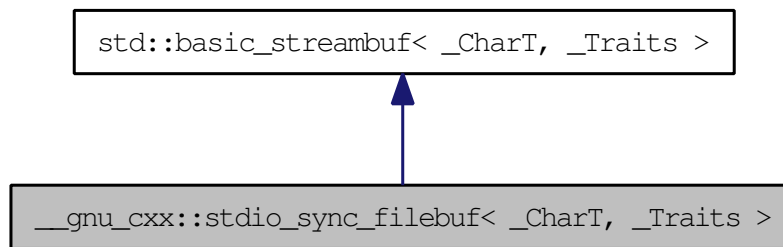
The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

## 5.72 `__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](#)

### 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` [sputc](#) (`char_type __c`)
- `streamsize` [sputn](#) (`const char_type *__s`, `streamsize __n`)
- void [stoss](#) ()
- `int_type` [sungetc](#) ()

- `template<>`  
`stdio_sync_filebuf< wchar_t >::int_type syncgetc ()`
- `template<>`  
`stdio_sync_filebuf< char >::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)`
- `template<>`  
`stdio_sync_filebuf< wchar_t >::int_type syncungetc (int_type __c)`
- `template<>`  
`stdio_sync_filebuf< char >::int_type syncungetc (int_type __c)`
- `template<>`  
`std::streamsize xsgetn (wchar_t * __s, std::streamsize __n)`
- `template<>`  
`std::streamsize xsgetn (char * __s, std::streamsize __n)`
- `template<>`  
`std::streamsize xspuwn (const wchar_t * __s, std::streamsize __n)`
- `template<>`  
`std::streamsize xspuwn (const char * __s, std::streamsize __n)`

## 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 pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
- `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 seekpos (pos_type, 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 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 ()`
- `int_type syncgetc ()`
- `int_type syncputc (int_type __c)`

- [int\\_type syncungetc](#) ([int\\_type](#) \_\_c)
- virtual [int\\_type uflow](#) ()
- virtual [int\\_type underflow](#) ()
- virtual [std::streamsize xsgetn](#) ([char\\_type](#) \*\_\_s, [std::streamsize](#) \_\_n)
- virtual [std::streamsize xspu](#) (const [char\\_type](#) \*\_\_s, [std::streamsize](#) \_\_n)

## 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 >`
- 
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
  - `locale getloc () const`
  - `locale pubimbue (const locale &__loc)`
  - `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)`
  - `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
  - `int pubsync ()`



- `char_type * eback () const`
- `char_type * egptr () const`
- `char_type * eptr () const`
- `char_type * gptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `locale _M_buf_locale`
- `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`

### 5.72.1 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 55 of file `stdio_sync_filebuf.h`.

### 5.72.2 Member Typedef Documentation

```
5.72.2.1 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 133 of file `streambuf`.

**5.72.2.2** `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 59 of file `stdio_sync_filebuf.h`.

**5.72.2.3** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::int_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 61 of file `stdio_sync_filebuf.h`.

**5.72.2.4** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::off_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::off_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 63 of file `stdio_sync_filebuf.h`.

**5.72.2.5** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::pos_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pos_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 62 of file `stdio_sync_filebuf.h`.

**5.72.2.6** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef _Traits __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::traits_type`

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 60 of file `stdio_sync_filebuf.h`.

### 5.72.3 Member Function Documentation

**5.72.3.1** `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 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.72.3.2** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr() const [inline, protected, 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 466 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.72.3.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eptr () const [inline,  
protected, 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 513 of file `streambuf`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.72.3.4** `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 87 of file `stdio_sync_filebuf.h`.

**5.72.3.5** `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 476 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.72.3.6** `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 222 of file `streambuf`.

**5.72.3.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr() const` [`inline`,  
`protected`, `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 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.72.3.8** `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 554 of file `streambuf`.

**5.72.3.9** `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 262 of file `streambuf`.

**5.72.3.10** `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*

## 5.72 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

1007

*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 140 of file `stdio_sync_filebuf.h`.

```
5.72.3.11 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 115 of file `stdio_sync_filebuf.h`.

**5.72.3.12** `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 507 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_filebuf< _CharT, _Traits >::sync()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.72.3.13** `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 523 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()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.72.3.14** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr () const [inline,  
protected, 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 510 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_filebuf<_CharT, _Traits>::sync()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.72.3.15** `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 205 of file `streambuf`.

**5.72.3.16** `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]`

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 239 of file `streambuf`.

**5.72.3.17** `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]`

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 244 of file `streambuf`.

**5.72.3.18** `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 235 of file `streambuf`.

**5.72.3.19** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync () [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 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.72.3.20** `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 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

**5.72.3.21** `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 580 of file `streambuf`.

**5.72.3.22** `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 592 of file `streambuf`.

**5.72.3.23** `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 569 of file `streambuf`.

**5.72.3.24** `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 487 of file `streambuf`.

**5.72.3.25** `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* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.72.3.26** `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 316 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()`.

**5.72.3.27** `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 335 of file `streambuf`.

**5.72.3.28** `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 627 of file `streambuf`.

**5.72.3.29** `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 276 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()`.

**5.72.3.30** `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 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.72.3.31** `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 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_-  
iterator<_CharT, _Traits>::operator=()`.

**5.72.3.32** `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 `xspun(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.72.3.33** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stossc () [inline,  
inherited]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.72.3.34** `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 `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.



**5.72.3.35** `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 159 of file `stdio_sync_filebuf.h`.

**5.72.3.36** `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 107 of file `stdio_sync_filebuf.h`.

**5.72.3.37** `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 100 of file `stdio_sync_filebuf.h`.

```
5.72.3.38 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 >`.

**5.72.3.39** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual std::streamsize __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xsputn(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>`.

## 5.72.4 Member Data Documentation

**5.72.4.1** `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected, inherited]`

Current [locale](#) setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

**5.72.4.2** `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 180 of file `streambuf`.

**5.72.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, 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 181 of file `streambuf`.

**5.72.4.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, 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 182 of file `streambuf`.

**5.72.4.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, 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 183 of file `streambuf`.

5.72.4.6 `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_cur  
[protected, 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 184 of file `streambuf`.

5.72.4.7 `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_end  
[protected, 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.

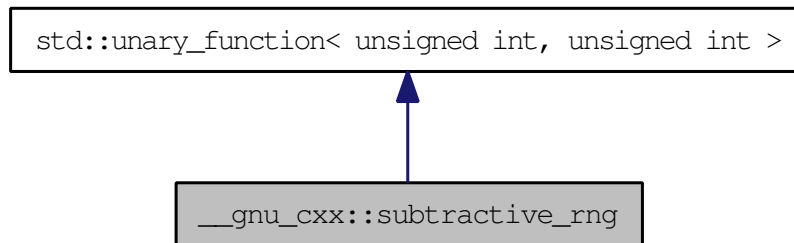
Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

## 5.73 `__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](#) ()
- [subtractive\\_rng](#) (unsigned int `__seed`)
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

#### 5.73.1 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 348 of file `ext/functional`.

#### 5.73.2 Member Typedef Documentation

- 5.73.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.73.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

### 5.73.3 Constructor & Destructor Documentation

**5.73.3.1** `__gnu_cxx::subtractive_rng::subtractive_rng (unsigned int __seed)  
[inline]`

Ctor allowing you to initialize the seed.

Definition at line 390 of file `ext/functional`.

**5.73.3.2** `__gnu_cxx::subtractive_rng::subtractive_rng () [inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 394 of file `ext/functional`.

### 5.73.4 Member Function Documentation

**5.73.4.1** `unsigned int __gnu_cxx::subtractive_rng::operator() (unsigned int  
__limit) [inline]`

Returns a number less than the argument.

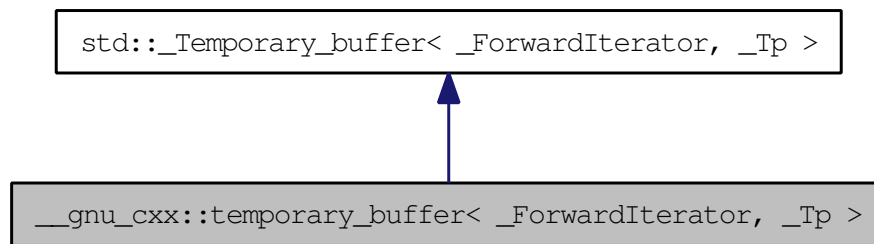
Definition at line 359 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.74 `__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**



### 5.74.1 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 182 of file `ext/memory`.

### 5.74.2 Constructor & Destructor Documentation

```
5.74.2.1 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 185 of file `ext/memory`.

```
5.74.2.2 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 189 of file `ext/memory`.

### 5.74.3 Member Function Documentation

**5.74.3.1** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  
[inline, inherited]`

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

**5.74.3.2** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  
[inline, inherited]`

As per Table mumble.

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

**5.74.3.3** `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 146 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

**5.74.3.4** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  
[inline, inherited]`

As per Table mumble.

Definition at line 141 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](#)

## 5.75 `__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 `allocate` (`size_type __n`, `std::allocator< void >::const_pointer hint=0`)
- void `check_allocated` (`void *p`, `size_t size`)
- void `check_allocated` (`size_type __n`)
- void `check_allocated` (`pointer __p`, `size_type __n`)
- template<typename... \_Args>  
 void `construct` (`pointer __p`, `_Args &&... __args`)
- void `construct` (`pointer __p`, `const value_type &val`)
- 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`)

### 5.75.1 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.

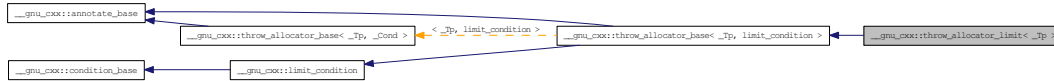
Definition at line 598 of file throw\_allocator.h.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

## 5.76 `__gnu_cxx::throw_allocator_limit< _Tp >` Struct Template Reference

Allocator throwing via limit condition. 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

- template<typename \_Tp1 >  
**throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)< \_Tp1 > &) throw ()
- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) throw ()
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_type \_\_n)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const value\_type &val)
- 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 & 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 ()`

### 5.76.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_limit< _Tp >`

Allocator throwing via limit condition.

Definition at line 682 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.77 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

### 5.77 `__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

- template<typename \_Tp1 >  
**throw\_allocator\_random** (const [throw\\_allocator\\_random](#)<\_Tp1 > &) throw ()
- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) throw ()
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_type \_\_n)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **construct** (pointer \_\_p, \_Args &&... \_\_args)
- void **construct** (pointer \_\_p, const value\_type &val)
- 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** ()

### 5.77.1 Detailed Description

**template<typename \_Tp> struct \_\_gnu\_cxx::throw\_allocator\_random< \_Tp >**

Allocator throwing via random condition.

Definition at line 701 of file `throw_allocator.h`.

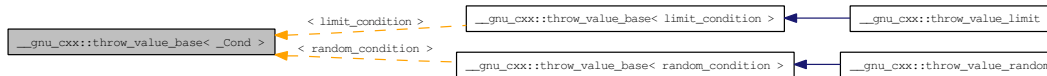
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)



## 5.78 `__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 `std::size_t __i`)
- `throw_value_base` (const `throw_value_base` &`__v`)
- `throw_value_base` & `operator++` ()
- `throw_value_base` & `operator=` (const `throw_value_base` &`__v`)

### Public Attributes

- `std::size_t __M_i`

#### 5.78.1 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.

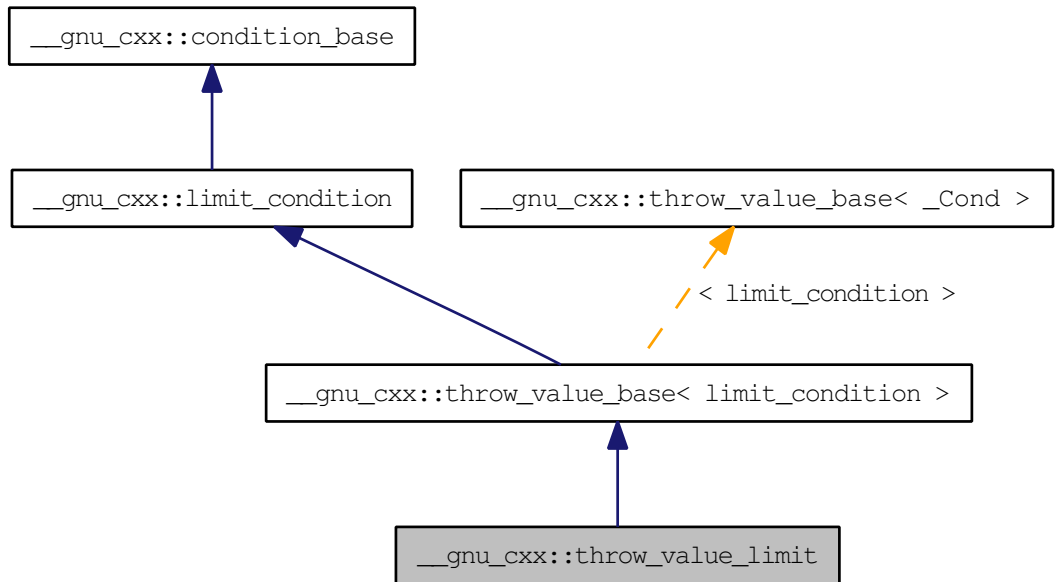
Definition at line 453 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.79 `__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 std::size\_t \_\_i)
- `throw_value_limit` (const `throw_value_limit` &\_\_other)
- `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`

**5.79.1 Detailed Description**

Type throwing via limit condition.

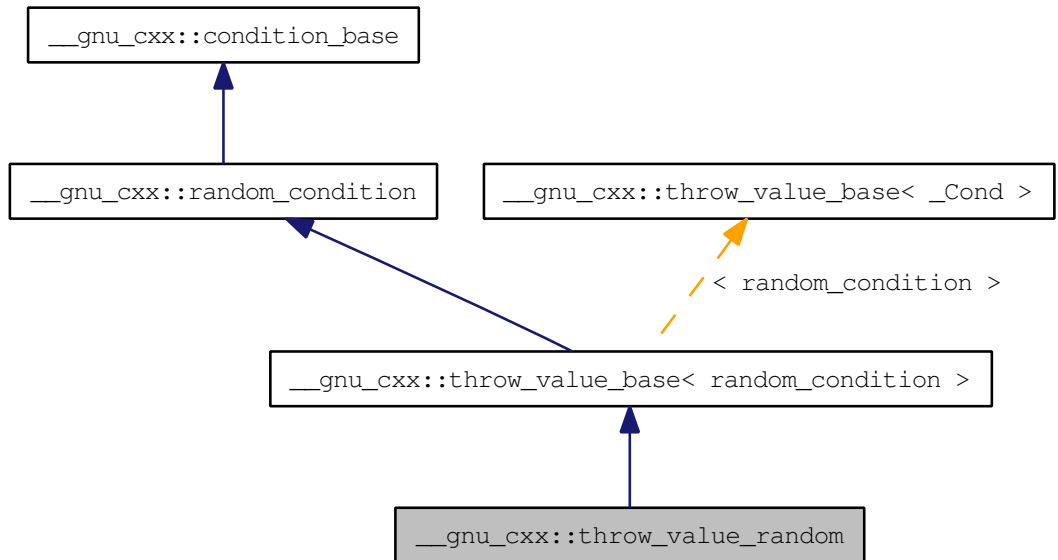
Definition at line 559 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.80 `__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 std::size\_t \_\_i)
- `throw_value_random` (const `throw_value_random` &\_\_other)
- `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`

### 5.80.1 Detailed Description

Type throwing via random condition.

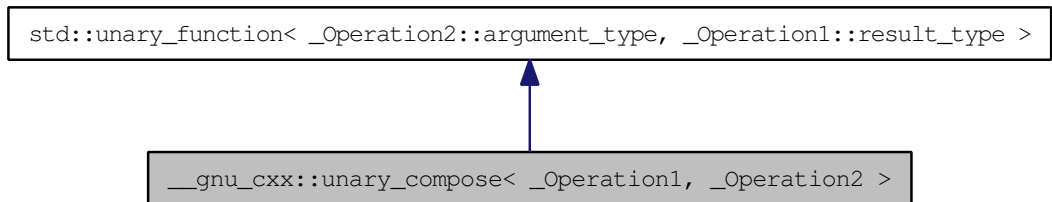
Definition at line 574 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.81 `__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

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**

#### 5.81.1 Detailed Description

```
template<class _Operation1, class _Operation2> class __gnu_cxx::unary_compose< _Operation1, _Operation2 >
```

An [SGI extension](#) .

Definition at line 125 of file `ext/functional`.

## 5.81.2 Member Typedef Documentation

**5.81.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function<_Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.81.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function<_Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.82 `__gnu_debug::__is_same< _Type1, _Type2 >` Struct Template Reference

### Static Public Attributes

- static const bool value

### 5.82.1 Detailed Description

```
template<typename _Type1, typename _Type2> struct __gnu_debug::__is_ -
same< _Type1, _Type2 >
```

Determine if the two types are the same.

Definition at line 43 of file `formatter.h`.

The documentation for this struct was generated from the following file:

- [formatter.h](#)



### 5.83 `__gnu_debug::_After_nth_from<_Iterator>` Class Template Reference

## 5.83 `__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

#### 5.83.1 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 63 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.84 `__gnu_debug::_Not_equal_to< _Type >` Class Template Reference

### Public Member Functions

- `_Not_equal_to` (const `_Type` &\_\_v)
- `bool operator()` (const `_Type` &\_\_x) const

### 5.84.1 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`.

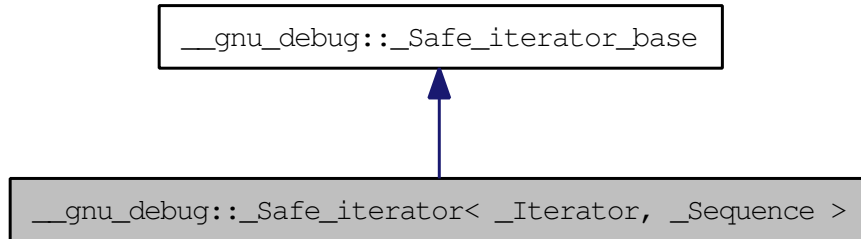
Definition at line 48 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.85 `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >` Class Template Reference

Safe iterator wrapper. Inheritance diagram for `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >`:



### Public Types

- typedef `_Iterator` **\_Base\_iterator**
- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::iterator_category` **iterator\_category**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value\_type**

### Public Member Functions

- `template<typename _MutableIterator >`  
`__Safe_iterator` (const `__Safe_iterator`< `_MutableIterator`, `typename __gnu_cxx::__enable_if<(std::__are_same< _MutableIterator, typename _Sequence::iterator::_Base_iterator >::__value), _Sequence >::__type >` &`_x`)
- `__Safe_iterator` (const `__Safe_iterator` &`_x`)
- `__Safe_iterator` (const `_Iterator` &`_i`, const `_Sequence` \*`_seq`)
- `__Safe_iterator` ()
- `__attribute__((__pure__))` `bool` `_M_can_compare`(const `__Safe_iterator_base` &`_x`) const throw ()
- `__attribute__((__pure__))` `bool` `_M_singular`() const throw ()
- `void` `_M_attach` (`__Safe_sequence_base` \*`_seq`, `bool` `__constant`)
- `void` `_M_attach` (const `_Sequence` \*`_seq`)
- `void` `_M_attach_single` (`__Safe_sequence_base` \*`_seq`, `bool` `__constant`) throw ()
- `void` `_M_attach_single` (const `_Sequence` \*`_seq`)

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_advance (const difference_type & __n) const`
- `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 ()`
- `void _M_invalidate_single ()`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `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++ (int)`
- `_Safe_iterator & operator++ ()`
- `_Safe_iterator & operator+= (const difference_type & __n)`
- `_Safe_iterator operator- (const difference_type & __n) const`
- `_Safe_iterator operator-- (int)`
- `_Safe_iterator & operator-- ()`
- `_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, 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)`
- `template<typename _Iterator1 , typename _Iterator2 >`  
`static std::pair< difference_type, _Distance_precision > _M_get_distance`  
`(const _Iterator1 & __lhs, const _Iterator2 & __rhs)`

## 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 ()

### 5.85.1 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 63 of file `safe_iterator.h`.

### 5.85.2 Constructor & Destructor Documentation

5.85.2.1 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator () [inline]`

#### Postcondition:

the iterator is singular and unattached

Definition at line 99 of file `safe_iterator.h`.

5.85.2.2 `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 108 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

```
5.85.2.3 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 119 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

```
5.85.2.4 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::_Base_iterator >::_value), _Sequence
 >::_type > & __x) [inline]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 136 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.85.3 Member Function Documentation**

```
5.85.3.1 __gnu_debug::_Safe_iterator_base::_attribute__ ((__pure__)) 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.

**5.85 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference** **1047**

---

**5.85.3.2** `__gnu_debug::Safe_iterator_base::__attribute__((__pure__)) const throw ()` **[inherited]**

Is this iterator singular?

**5.85.3.3** `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()`.

**5.85.3.4** `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_attach (const _Sequence * __seq)` **[inline]**

Attach iterator to the given sequence.

Definition at line 321 of file `safe_iterator.h`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=()`.

**5.85.3.5** `void __gnu_debug::Safe_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()` **[inherited]**

Likewise, but not thread-safe.

**5.85.3.6** `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_attach_single (const _Sequence * __seq)` **[inline]**

Likewise, but not thread-safe.

Definition at line 329 of file `safe_iterator.h`.

**5.85.3.7** `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`.

**5.85.3.8** `template<typename _Iterator, typename _Sequence>`  
`bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence`  
`>::_M_dereferenceable () const [inline]`

Is the iterator dereferenceable?

Definition at line 345 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`.

Referenced by `__gnu_debug::_check_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`.

**5.85.3.9** `void __gnu_debug::_Safe_iterator_base::_M_detach ()`  
`[inherited]`

Detach the iterator for whatever sequence it is attached to, if any.

**5.85.3.10** `void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw`  
`() [inherited]`

Likewise, but not thread-safe.

**5.85.3.11** `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 375 of file `safe_iterator.h`.

**5.85.3.12** `__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_-`  
`base::_M_get_mutex () throw () [protected,`  
`inherited]`

For use in `_Safe_iterator`.



**5.85 \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence > Class Template Reference** **1049**

---

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_invalidate().

**5.85.3.13** `template<typename _Iterator, typename _Sequence>  
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
>::_M_incrementable() const [inline]`

Is the iterator incrementable?

Definition at line 350 of file safe\_iterator.h.

References \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_dereferenceable().

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::operator++().

**5.85.3.14** `template<typename _Iterator , typename _Sequence >  
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
>::_M_invalidate() [inline]`

Invalidate the iterator, making it singular.

Definition at line 106 of file safe\_iterator.tcc.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_get\_mutex(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_invalidate\_single().

**5.85.3.15** `template<typename _Iterator , typename _Sequence >  
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
>::_M_invalidate_single() [inline]`

Likewise, but not thread-safe.

Definition at line 115 of file safe\_iterator.tcc.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_const\_iterators, \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_iterators, \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next, \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_sequence, \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_version, and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::base().

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_invalidate().

**5.85.3.16** `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 400 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

**5.85.3.17** `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 404 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**5.85.3.18** `template<typename _Iterator, typename _Sequence> _Iterator  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base () const  
[inline]`

Return the underlying iterator.

Definition at line 311 of file safe\_iterator.h.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.85.3.19** `template<typename _Iterator, typename _Sequence>  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator  
_Iterator () const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 317 of file safe\_iterator.h.

**5.85 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference** **1051**

---

**5.85.3.20** `template<typename _Iterator, typename _Sequence> reference  
__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator* ()  
const [inline]`

Iterator dereference.

**Precondition:**

iterator is dereferenceable

Definition at line 175 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

**5.85.3.21** `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 218 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

**5.85.3.22** `template<typename _Iterator, typename _Sequence>  
_Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence  
>::operator++ () [inline]`

Iterator preincrement.

**Precondition:**

iterator is incrementable

Definition at line 204 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

**5.85.3.23** `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 248 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.85.3.24** `template<typename _Iterator, typename _Sequence>  
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
>::operator-- () [inline]`

Iterator predecrement.

**Precondition:**

iterator is decrementable

Definition at line 234 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.85.3.25** `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 190 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _-  
Iterator, _Sequence >::_M_dereferenceable()`.

## 5.85 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1053

---

**5.85.3.26** `template<typename _Iterator, typename _Sequence>  
_Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence  
>::operator= (const _Safe_iterator<_Iterator, _Sequence> & __x)  
[inline]`

Copy assignment.

Reimplemented from [\\_\\_gnu\\_debug::Safe\\_iterator\\_base](#).

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

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_attach()`, and `__gnu_debug::Safe_iterator_base::M_sequence`.

### 5.85.4 Member Data Documentation

**5.85.4.1** `_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_iterator<_Iterator, _Sequence>::M_invalidate_single()`.

**5.85.4.2** `_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`.

**5.85.4.3** `_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_iterator<_Iterator, _Sequence>::M_invalidate_single()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`, `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_iter()`, `__gnu_debug::Safe_iterator_`

base::\_Safe\_iterator\_base(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::operator=().

#### 5.85.4.4 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.

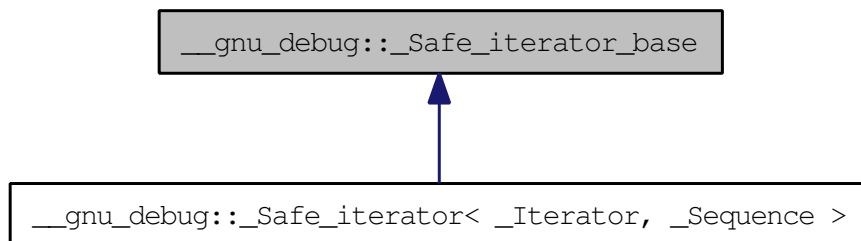
Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_invalidate\_single().

The documentation for this class was generated from the following files:

- [safe\\_iterator.h](#)
- [safe\\_iterator.tcc](#)

## 5.86 `__gnu_debug::_Safe_iterator_base` Class Reference

Basic functionality for a *safe* iterator. Inheritance diagram for `__gnu_debug::_Safe_iterator_base`:



### Public Member Functions

- `__attribute__((__pure__)) bool _M_can_compare(const _Safe_iterator_base &_x) const` throw ()
- `__attribute__((__pure__)) bool _M_singular() const` throw ()
- `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`
- `void _M_detach() throw ()`
- `void _M_detach_single() 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(const _Safe_iterator_base &)`
- `_Safe_iterator_base(const _Safe_iterator_base &_x, bool __constant)`
- `_Safe_iterator_base(const _Safe_sequence_base *__seq, bool __constant)`
- `_Safe_iterator_base()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw ()`
- `_Safe_iterator_base & operator=(const _Safe_iterator_base &)`

### 5.86.1 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`.

### 5.86.2 Constructor & Destructor Documentation

#### 5.86.2.1 `__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`.

#### 5.86.2.2 `__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()`.

#### 5.86.2.3 `__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`.



### 5.86.3 Member Function Documentation

#### 5.86.3.1 `__gnu_debug::_Safe_iterator_base::_attribute__ ((__pure__)) const throw ()`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

#### 5.86.3.2 `__gnu_debug::_Safe_iterator_base::_attribute__ ((__pure__)) const throw ()`

Is this iterator singular?

#### 5.86.3.3 `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()`.

#### 5.86.3.4 `void __gnu_debug::_Safe_iterator_base::_M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`

Likewise, but not thread-safe.

#### 5.86.3.5 `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`.

#### 5.86.3.6 `void __gnu_debug::_Safe_iterator_base::_M_detach ()`

Detach the iterator for whatever sequence it is attached to, if any.

#### 5.86.3.7 `void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw ()`

Likewise, but not thread-safe.

### 5.86.3.8 `__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw ()` [protected]

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

## 5.86.4 Member Data Documentation

### 5.86.4.1 `_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`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

### 5.86.4.2 `_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`.

### 5.86.4.3 `_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_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

### 5.86.4.4 `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 `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.87 `__gnu_debug::_Safe_sequence< _Sequence >` Class Template Reference

Base class for constructing a *safe* sequence type that tracks iterators that reference it. Inheritance diagram for `__gnu_debug::_Safe_sequence< _Sequence >`:



### Public Member Functions

- `void _M_invalidate_all () const`
- `template<typename _Predicate > void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Iterator > void _M_transfer_iter (const _Safe_iterator< _Iterator, _Sequence > &__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)`

## 5.87 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference 061

### 5.87.1 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 97 of file `safe_sequence.h`.

### 5.87.2 Member Function Documentation

**5.87.2.1** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.87.2.2** `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`.

**5.87.2.3** `__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_iter()`.

**5.87.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`  
**[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

**5.87.2.5** `template<typename Sequence> template<typename Predicate>`  
`void __gnu_debug::Safe_sequence<Sequence>::_M_invalidate_if`  
`(Predicate __pred) [inline]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

Definition at line 121 of file `safe_sequence.h`.

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`.

**5.87.2.6** `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).

**5.87.2.7** `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.

## 5.87 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference 063

**5.87.2.8** `template<typename _Sequence> template<typename _Iterator >`  
`void __gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter`  
`(const _Safe_iterator<_Iterator, _Sequence > & __x) [inline]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

Definition at line 154 of file `safe_sequence.h`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, `__gnu_debug::_Safe_sequence_base::_M_iterators`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::base()`.

### 5.87.3 Member Data Documentation

**5.87.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.87.3.2** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.87.3.3** `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

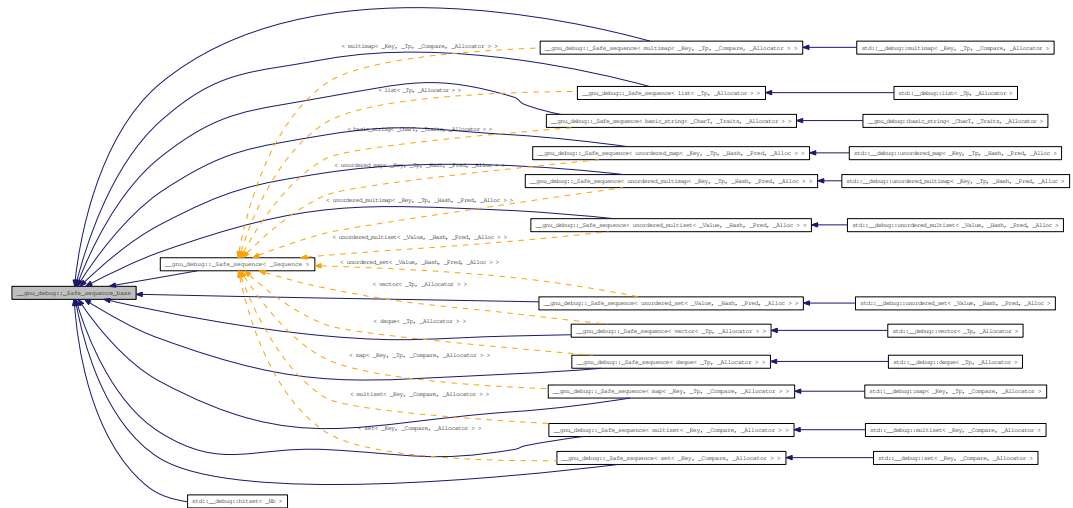
Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.88 `__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_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)`



### 5.88.1 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 159 of file `safe_base.h`.

### 5.88.2 Constructor & Destructor Documentation

**5.88.2.1** `__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 179 of file `safe_base.h`.

### 5.88.3 Member Function Documentation

**5.88.3.1** `void __gnu_debug::_Safe_sequence_base::M_detach_all ()`  
[protected]

Detach all iterators, leaving them singular.

**5.88.3.2** `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`.

**5.88.3.3** `__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_iter()`.

#### 5.88.3.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const` [`inline`]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

#### 5.88.3.5 `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).

#### 5.88.3.6 `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.

### 5.88.4 Member Data Documentation

#### 5.88.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

#### 5.88.4.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

#### 5.88.4.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` `[mutable]`

The container version number. This number may never be 0.

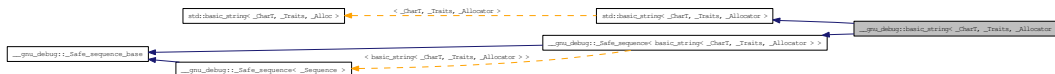
Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.89 `__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 `__gnu_debug::Safe_iterator<typename _Base::const_iterator, basic_string>` **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, basic_string>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

### Public Member Functions

- `basic_string` (`std::initializer_list<_CharT>` `__l`, `const _Allocator &__a=_Allocator()`)
- `basic_string` (`basic_string &&__str`)
- `template<typename _InputIterator>`  
`basic_string` (`_InputIterator __begin`, `_InputIterator __end`, `const _Allocator &__a=_Allocator()`)
- `basic_string` (`size_type __n`, `_CharT __c`, `const _Allocator &__a=_Allocator()`)
- `basic_string` (`const _CharT *__s`, `const _Allocator &__a=_Allocator()`)
- `basic_string` (`const _CharT *__s`, `size_type __n`, `const _Allocator &__a=_Allocator()`)

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1069

---

- `basic_string` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n= `Base::npos`, const `_Allocator` &\_\_a=`_Allocator`())
- `basic_string` (const `basic_string` &\_\_str)
- `basic_string` (const `_Base` &\_\_base)
- `basic_string` (const `_Allocator` &\_\_a=`_Allocator`())
- `~basic_string` ()
- const `_Base` & `_M_base` () const
- `_Base` & `_M_base` ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `basic_string`< `_CharT`, `_Traits`, `_Allocator` > > &\_\_x)
- `basic_string` & `append` (initializer\_list< `_CharT` > \_\_l)
- `basic_string` & `append` (size\_type \_\_n, `_CharT` \_\_c)
- `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 `basic_string` &\_\_str)
- template<typename `_InputIterator` >  
`basic_string` & `append` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `basic_string` & `append` (size\_type \_\_n, `_CharT` \_\_c)
- `basic_string` & `append` (const `_CharT` \*\_\_s)
- `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 `basic_string` &\_\_str)
- `basic_string` & `assign` (size\_type \_\_n, `_CharT` \_\_c)
- `basic_string` & `assign` (const `_CharT` \*\_\_s, size\_type \_\_n)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & `assign` (`basic_string` &&\_\_str)
- `basic_string` & `assign` (const `basic_string` &\_\_str)
- `basic_string` & `assign` (std::initializer\_list< `_CharT` > \_\_l)
- template<typename `_InputIterator` >  
`basic_string` & `assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `basic_string` & `assign` (size\_type \_\_n, `_CharT` \_\_c)
- `basic_string` & `assign` (const `_CharT` \*\_\_s)
- `basic_string` & `assign` (const `_CharT` \*\_\_s, size\_type \_\_n)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & `assign` (`basic_string` &&\_\_x)
- `basic_string` & `assign` (const `basic_string` &\_\_x)
- reference at (size\_type \_\_n)

- `const_reference at (size_type __n) const`
- `const_iterator begin () const`
- `iterator begin ()`
- `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, size_type __n2) const`
- `int compare (size_type __pos, size_type __n1, const _CharT *__s) 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 __n, const basic_string &__str) const`
- `int compare (const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const _CharT *__s, 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 basic_string &__str, size_type __pos2, size_type __n2) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str) const`
- `int compare (const basic_string &__str) 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`
- `bool empty () const`
- `const_iterator end () const`
- `iterator end ()`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `basic_string & erase (size_type __pos=0, size_type __n=npos)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `basic_string & erase (size_type __pos=0, size_type __n=_Base::npos)`
- `size_type find (_CharT __c, size_type __pos=0) const`
- `size_type find (const _CharT *__s, 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 (_CharT __c, size_type __pos=0) const`

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1071**

---

- `size_type find` (const `_CharT * __s`, `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_first_not_of` (`_CharT __c`, `size_type __pos=0`) const
- `size_type find_first_not_of` (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` (const `basic_string & __str`, `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=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 `basic_string & __str`, `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=0`) const
- `size_type find_first_of` (const `_CharT * __s`, `size_type __pos`, `size_type __n`) const
- `size_type find_first_of` (const `basic_string & __str`, `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=0`) const
- `size_type find_first_of` (const `_CharT * __s`, `size_type __pos`, `size_type __n`) 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=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` (const `basic_string & __str`, `size_type __pos=npos`) 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=_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 `basic_string & __str`, `size_type __pos=_Base::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=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=npos`) const
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=_Base::npos`) const

- size\_type **find\_last\_of** (const \_CharT \*\_\_s, 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 [basic\\_string](#) &\_\_str, size\_type \_\_pos=[Base::npos](#)) const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_p, \_CharT \_\_c)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- void **insert** (iterator \_\_p, initializer\_list< \_CharT > \_\_l)
- void **insert** (iterator \_\_p, InputIterator \_\_beg, InputIterator \_\_end)
- void **insert** (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **insert** (iterator \_\_p, [std::initializer\\_list](#)< \_CharT > \_\_l)
- template<typename InputIterator >  
void **insert** (iterator \_\_p, InputIterator \_\_first, InputIterator \_\_last)
- void **insert** (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator **insert** (iterator \_\_p, \_CharT \_\_c)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- size\_type **length** () const
- size\_type **max\_size** () const
- [basic\\_string](#) & **operator+=** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **operator+=** ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & **operator+=** (\_CharT \_\_c)
- [basic\\_string](#) & **operator+=** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **operator+=** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **operator=** ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & **operator=** ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & **operator=** (\_CharT \_\_c)
- [basic\\_string](#) & **operator=** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **operator=** (const [basic\\_string](#) &\_\_str)
- reference **operator[ ]** (size\_type \_\_pos)
- const\_reference **operator[ ]** (size\_type \_\_pos) const
- reference **operator[ ]** (size\_type \_\_pos)



## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1073

- `const_reference operator[]` (size\_type \_\_pos) const
- `void push_back` (\_CharT \_\_c)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, initializer\_list< \_CharT > \_\_l)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const basic\_string & \_\_str)
- `basic_string & replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- `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` (size\_type \_\_pos, size\_type \_\_n, const basic\_string & \_\_str)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, std::initializer\_list< \_CharT > \_\_l)
- `template<typename _InputIterator > basic_string & replace` (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_j1, \_InputIterator \_\_j2)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string & replace` (iterator \_\_i1, iterator \_\_i2, const basic\_string & \_\_str)
- `basic_string & replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- `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** (size\_type \_\_pos1, size\_type \_\_n1, 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** (`_CharT` \_\_c, size\_type \_\_pos=`npos`) const
- size\_type **rfind** (const `_CharT` \*\_\_s, 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** (`_CharT` \_\_c, size\_type \_\_pos=`_Base::npos`) const
- size\_type **rfind** (const `_CharT` \*\_\_s, size\_type \_\_pos=`_Base::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=`_Base::npos`) const
  
- void **shrink\_to\_fit** ()
- size\_type **size** () const
- `basic_string` **substr** (size\_type \_\_pos=0, size\_type \_\_n=`npos`) const
- `basic_string` **substr** (size\_type \_\_pos=0, size\_type \_\_n=`_Base::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)

### 5.89.1 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`.

### 5.89.2 Constructor & Destructor Documentation

**5.89.2.1** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::basic_string(const _Allocator & __a = _Allocator()) [inline, explicit]`

Construct an empty string using allocator *a*.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 73 of file `debug/string`.

**5.89.2.2** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::basic_string(const _CharT * __s, const _Allocator & __a = _Allocator()) [inline]`

Construct string as copy of a C string.

#### Parameters:

*s* Source C string.

*a* Allocator to use (default is default allocator).

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 99 of file `debug/string`.

```

5.89.2.3 template<typename _CharT, typename _Traits =
std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>> template<typename _InputIterator
> __gnu_debug::basic_string<_CharT, _Traits, _Allocator
>::basic_string(_InputIterator __beg, _InputIterator __end, const
_Allocator & __a = _Allocator()) [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).

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 109 of file debug/string.

```

5.89.2.4 template<typename _CharT, typename _Traits =
std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>> __gnu_debug::basic_string<_CharT,
_Traits, _Allocator>::basic_string(std::initializer_list<_CharT>
__l, const _Allocator & __a = _Allocator()) [inline]

```

Construct string from an initializer list.

**Parameters:**

*l* [std::initializer\\_list](#) of characters.

*a* Allocator to use (default is default allocator).

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 119 of file debug/string.

```

5.89.2.5 template<typename _CharT, typename _Traits =
std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>> __gnu_debug::basic_string<_CharT,
_Traits, _Allocator>::~~basic_string() [inline]

```

Destroy the string instance.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 125 of file debug/string.

### 5.89.3 Member Function Documentation

**5.89.3.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.89.3.2** `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`.

**5.89.3.3** `__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_iter()`.

**5.89.3.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

**5.89.3.5** `void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_invalidate_if ( _Predicate __pred) [inline, inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.89.3.6** `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).

**5.89.3.7** `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.

**5.89.3.8** `void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator, basic_string< _CharT, _Traits, _Allocator > > & __x) [inline, inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

**5.89.3.9** `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 977 of file `basic_string.h`.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **1079**

---

**5.89.3.10 `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.

**5.89.3.11 `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.

**5.89.3.12 `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.

**5.89.3.13** `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.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::append()`.

**5.89.3.14** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> template<typename _InputIterator > basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator >::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.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator >`.

Definition at line 345 of file `debug/string`.

References `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `std::basic_string<_CharT, _Traits, _Allocator >::append()`.



**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **1081**

---

**5.89.3.15** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append(const _CharT * __s) [inline]`

Append a C string.

**Parameters:**

*s* The C string to append.

**Returns:**

Reference to this string.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 327 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::_M_invalidate_all()`, and `std::basic_string<_CharT, _Traits, _Allocator>::append()`.

**5.89.3.16** `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 1090 of file `basic_string.h`.

**5.89.3.17** `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.

**5.89.3.18** `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 1046 of file `basic_string.h`.

**5.89.3.19** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >` Class Template Reference 1083

---

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1026 of file `basic_string.h`.

**5.89.3.20** `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.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::assign()`.

**5.89.3.21** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator >::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.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator >`.

Definition at line 426 of file `debug/string`.

References `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `std::basic_string<_CharT, _Traits, _Allocator >::assign()`.

```

5.89.3.22 template<typename _CharT , typename _Traits =
std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>> template<typename _InputIterator
> basic_string& __gnu_debug::basic_string< _CharT, _Traits,
_Allocator >::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).

Reimplemented from [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 416 of file debug/string.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_invalidate\\_all\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::assign\(\)](#).

```

5.89.3.23 template<typename _CharT , typename _Traits = std::char_
traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
basic_string& __gnu_debug::basic_string< _CharT, _Traits,
_Allocator >::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.

Reimplemented from [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 398 of file debug/string.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_invalidate\\_all\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::assign\(\)](#).

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1085

---

### 5.89.3.24 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 865 of file `basic_string.h`.

### 5.89.3.25 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 846 of file `basic_string.h`.

**5.89.3.26** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> const_iterator __gnu_debug::basic_string< _CharT, _Traits, _Allocator >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Reimplemented from [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 176 of file debug/string.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::begin\(\)](#).

**5.89.3.27** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> iterator __gnu_debug::basic_string< _CharT, _Traits, _Allocator >::begin () [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Reimplemented from [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 172 of file debug/string.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::begin\(\)](#).

Referenced by [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::rend\(\)](#).

**5.89.3.28** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> const _CharT* __gnu_debug::basic_string< _CharT, _Traits, _Allocator >::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.

Reimplemented from [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 664 of file debug/string.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_invalidate\\_all\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >::c\\_str\(\)](#).

**5.89.3.29** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1087

---

Definition at line 758 of file `basic_string.h`.

**5.89.3.30** `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 665 of file `basic_string.h`.

**5.89.3.31** `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 673 of file `basic_string.h`.

**5.89.3.32** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> void` `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear() [inline]`

Erases the string, making it empty.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 227 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `std::basic_string<_CharT, _Traits, _Allocator>::clear()`.

**5.89.3.33** `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 *r1en* 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,r1en)`. 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.

**5.89.3.34** `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 *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.

**5.89.3.35** `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.



## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1089

---

*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.

**5.89.3.36** `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.

**5.89.3.37** `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 2129 of file `basic_string.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::compare()`.

```
5.89.3.38 template<typename _CharT , typename _Traits =
std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>> int __gnu_debug::basic_string< _CharT,
_Traits, _Allocator >::compare (const _CharT * __s) const
[inline]
```

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.

Reimplemented from `std::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 834 of file `debug/string`.

References `std::basic_string< _CharT, _Traits, _Allocator >::compare()`.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1091**

---

**5.89.3.39** `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.

**5.89.3.40** `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::crbegin` () `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 682 of file `basic_string.h`.

**5.89.3.41** `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 691 of file `basic_string.h`.

**5.89.3.42** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>`  
`const _CharT* __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data` () `const` [`inline`]

Return const pointer to contents. This is a handle to internal data. Do not modify or dire things may happen.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 672 of file debug/string.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_invalidate\\_all\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>::data\(\)](#).

**5.89.3.43** `bool std::basic_string<_CharT, _Traits, _Allocator>::empty() const [inline, inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 793 of file basic\_string.h.

**5.89.3.44** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> const_iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::end() const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 184 of file debug/string.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>::end\(\)](#).

**5.89.3.45** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::end() [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 180 of file debug/string.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>::end\(\)](#).

Referenced by [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>::rbegin\(\)](#).

**5.89.3.46** `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase(iterator __first, iterator __last) [inherited]`

Remove a range of characters.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1093

---

### 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.

### 5.89.3.47 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 1319 of file `basic_string.h`.

### 5.89.3.48 `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 1303 of file `basic_string.h`.

**5.89.3.49** `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`.

**5.89.3.50** `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 1781 of file `basic_string.h`.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **1095**

---

**5.89.3.51** `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 1767 of file `basic_string.h`.

**5.89.3.52** `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*.

**5.89.3.53** `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.

**5.89.3.54** `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 2020 of file basic\_string.h.

**5.89.3.55** `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.



## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1097

---

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*.

**5.89.3.56** `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 1991 of file `basic_string.h`.

**5.89.3.57** `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 1916 of file `basic_string.h`.

**5.89.3.58** `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 1897 of file `basic_string.h`.

**5.89.3.59** `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*.

**5.89.3.60** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1099

---

### 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 1869 of file `basic_string.h`.

### 5.89.3.61 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of(_CharT __c, size_type __pos = npos) 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`.

### 5.89.3.62 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of(const _CharT * __s, size_type __pos = npos) 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 2079 of file `basic_string.h`.

**5.89.3.63** `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*.

**5.89.3.64** `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 2050 of file `basic_string.h`.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1101**

---

**5.89.3.65** `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 1977 of file `basic_string.h`.

**5.89.3.66** `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 1958 of file `basic_string.h`.

**5.89.3.67** `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`.

**5.89.3.68** `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 1930 of file `basic_string.h`.

**5.89.3.69** `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 1739 of file `basic_string.h`.

**5.89.3.70** `iterator std::basic_string<_CharT, _Traits, _Allocator>::insert(iterator __p, _CharT __c)` [`inline`, `inherited`]

Insert one character.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1103

---

### 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 1279 of file `basic_string.h`.

### 5.89.3.71 `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 1262 of file `basic_string.h`.

**5.89.3.72** `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 1239 of file `basic_string.h`.

**5.89.3.73** `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.



## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1105

---

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.

**5.89.3.74** `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 1198 of file `basic_string.h`.

**5.89.3.75** `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 *pos*. 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 1176 of file `basic_string.h`.

**5.89.3.76** `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 1157 of file `basic_string.h`.

**5.89.3.77** `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()`.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1107

---

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 1146 of file `basic_string.h`.

**5.89.3.78** `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 1130 of file `basic_string.h`.

**5.89.3.79** `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 706 of file `basic_string.h`.

**5.89.3.80** `size_type std::basic_string<_CharT, _Traits, _Allocator>::max_size`  
( ) const [`inline`, `inherited`]

Returns the `size()` of the largest possible string.

Definition at line 711 of file `basic_string.h`.

**5.89.3.81** `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 880 of file `basic_string.h`.

**5.89.3.82** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::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.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 293 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::M_invalidate_all()`.

**5.89.3.83** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=(_CharT __c)` [`inline`]

Append a character.

**Parameters:**

*c* The character to append.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1109

---

### Returns:

Reference to this string.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 284 of file `debug/string`.

References `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

**5.89.3.84** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=(const _CharT * __s) [inline]`

Append a C string.

### Parameters:

*s* The C string to append.

### Returns:

Reference to this string.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 275 of file `debug/string`.

References `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

**5.89.3.85** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>> basic_string& __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=(std::initializer_list<_CharT> __l) [inline]`

Set value to string constructed from initializer list.

### Parameters:

*l* [std::initializer\\_list](#).

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#).

Definition at line 162 of file `debug/string`.

References `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

**5.89.3.86** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string< _CharT, _Traits, _Allocator >::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`.

Reimplemented from `std::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 145 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::M_invalidate_all()`.

**5.89.3.87** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> basic_string& __gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=(const _CharT * __s) [inline]`

Copy contents of *s* into this string.

**Parameters:**

*s* Source null-terminated string.

Reimplemented from `std::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 136 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::M_invalidate_all()`.

**5.89.3.88** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >` Class Template Reference 1111

---

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 825 of file `basic_string.h`.

### 5.89.3.89 `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 808 of file `basic_string.h`.

### 5.89.3.90 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> void` `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::push_back (_CharT __c)` `[inline]`

Append a single character.

#### Parameters:

*c* Character to append.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator >`.

Definition at line 356 of file `debug/string`.

References `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `std::basic_string<_CharT, _Traits, _Allocator >::push_back()`.

**5.89.3.91** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> const_reverse_iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator >::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.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 192 of file debug/string.

References [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >::end\(\)](#).

**5.89.3.92** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> reverse_iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator >::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.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 188 of file debug/string.

References [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >::end\(\)](#).

**5.89.3.93** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> const_reverse_iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator >::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.

Reimplemented from [std::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 200 of file debug/string.

References [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >::begin\(\)](#).



## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1113

---

**5.89.3.94** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> reverse_iterator __gnu_debug::basic_string<_CharT, _Traits, _Allocator>::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.

Reimplemented from `std::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 196 of file `debug/string`.

References `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::begin()`.

**5.89.3.95** `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 1615 of file `basic_string.h`.

**5.89.3.96** `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 1547 of file `basic_string.h`.

**5.89.3.97** `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 1524 of file `basic_string.h`.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **1115**

---

**5.89.3.98** `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 1503 of file `basic_string.h`.

**5.89.3.99** `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 1482 of file `basic_string.h`.

**5.89.3.100** `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 1464 of file `basic_string.h`.

**5.89.3.101** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1117

---

### 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 1446 of file `basic_string.h`.

### 5.89.3.102 `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 first *n* 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 1423 of file `basic_string.h`.

**5.89.3.103** `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.

**5.89.3.104** `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.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1119

---

### 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 1380 of file `basic_string.h`.

### 5.89.3.105 `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 1358 of file `basic_string.h`.

### 5.89.3.106 `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.

**5.89.3.107 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 738 of file `basic_string.h`.

**5.89.3.108 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*.



**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **1121**

---

**5.89.3.109** `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`.

**5.89.3.110** `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 1839 of file `basic_string.h`.

**5.89.3.111** `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*.

**5.89.3.112** `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 1811 of file `basic_string.h`.

**5.89.3.113** `void std::basic_string<_CharT, _Traits, _Allocator>::shrink_to_fit() [inline, inherited]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 744 of file `basic_string.h`.

**5.89.3.114** `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 700 of file `basic_string.h`.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1123

---

**5.89.3.115** `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 2111 of file `basic_string.h`.

**5.89.3.116** `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.

## 5.89.4 Member Data Documentation

**5.89.4.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

**5.89.4.2** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_`  
`iterators` **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.89.4.3** `unsigned int __gnu_debug::_Safe_sequence_base::_M_version`  
`[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

**5.89.4.4** `const size_type std::basic_string<_CharT, _Traits, _Allocator`  
`>::npos` **[static, inherited]**

Value returned by various member functions when they fail.

Definition at line 270 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

## 5.90 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

General reduction, using a binary operator.

### 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`

#### 5.90.1 Detailed Description

```
template<typename _BinOp> struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

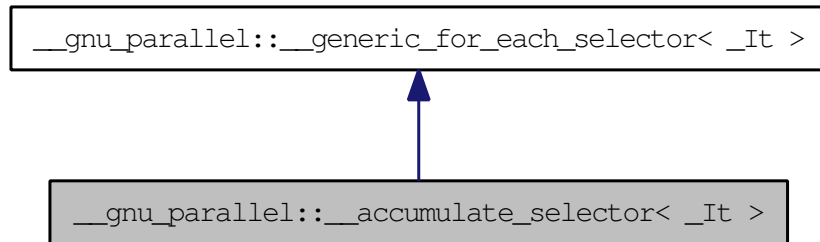
Definition at line 335 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.91 `__gnu_parallel::__accumulate_selector< _It >` Struct Template Reference

`std::accumulate()` selector. Inheritance diagram for `__gnu_parallel::__accumulate_selector< _It >`:



### Public Member Functions

- `template<typename _Op > std::iterator_traits< _It >::value_type operator() (_Op __o, _It __i)`

### Public Attributes

- `_It \_M\_finish\_iterator`

#### 5.91.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector< _It >`

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

#### 5.91.2 Member Function Documentation

**5.91.2.1** `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.

## 5.91 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

### Parameters:

- `__o` Operator (unused).
- `__i` iterator referencing object.

### Returns:

The current value.

Definition at line 216 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()`).

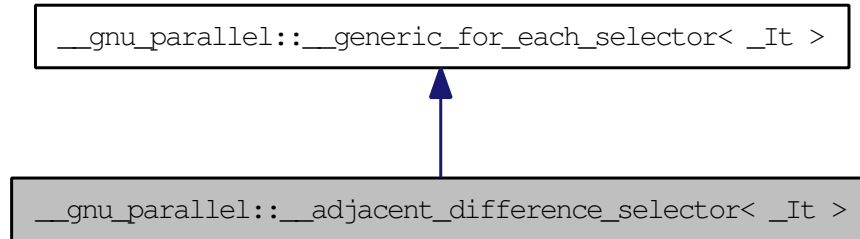
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.92 `__gnu_parallel::__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`

### 5.92.1 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`.

### 5.92.2 Member Data Documentation

**5.92.2.1** `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()`).



## 5.92 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference 1129

---

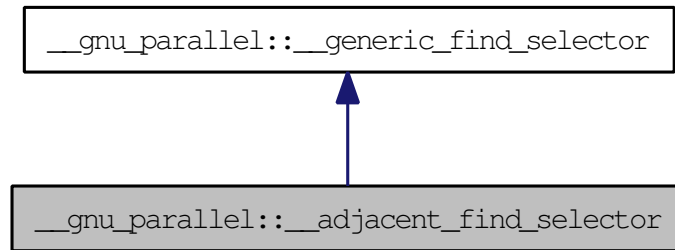
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.93 `__gnu_parallel::__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)`

### 5.93.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

### 5.93.2 Member Function Documentation

**5.93.2.1** `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`.

```
5.93.2.2 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.

**Parameters:**

`__i1` \_Iterator on first sequence.

`__i2` \_Iterator on second sequence (unused).

`__pred` Find predicate.

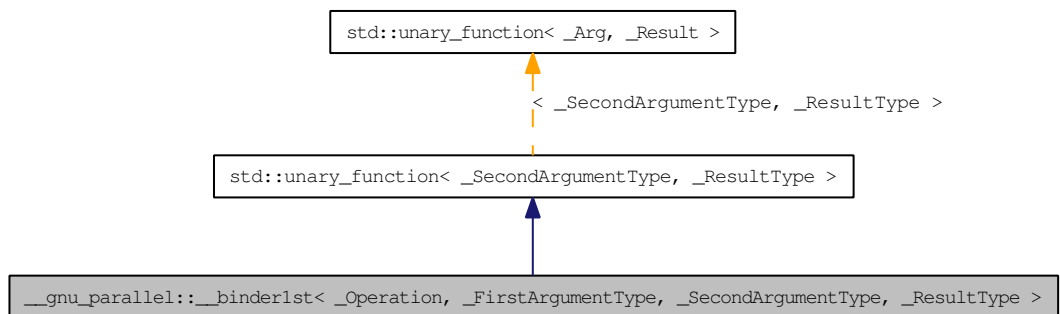
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.94 `__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`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`
- `_ResultType operator()` (`const _SecondArgumentType &__x`)

### Protected Attributes

- `_Operation` `_M_op`
- `_FirstArgumentType` `_M_value`

### 5.94.1 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`.

### 5.94.2 Member Typedef Documentation

**5.94.2.1** `typedef _SecondArgumentType std::unary_function< _SecondArgumentType , _ResultType >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.94.2.2** `typedef _ResultType std::unary_function< _SecondArgumentType , _ResultType >::result_type [inherited]`

`result_type` is the return type

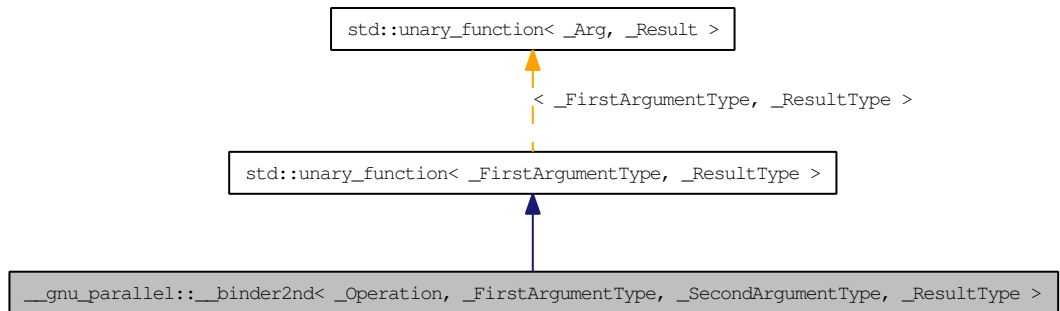
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.95 `__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`

### 5.95.1 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`.

### 5.95.2 Member Typedef Documentation

**5.95.2.1** `typedef _FirstArgumentType std::unary_function<  
_FirstArgumentType , _ResultType >::argument_type  
[inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.95.2.2** `typedef _ResultType std::unary_function< _FirstArgumentType ,  
_ResultType >::result_type [inherited]`

`result_type` is the return type

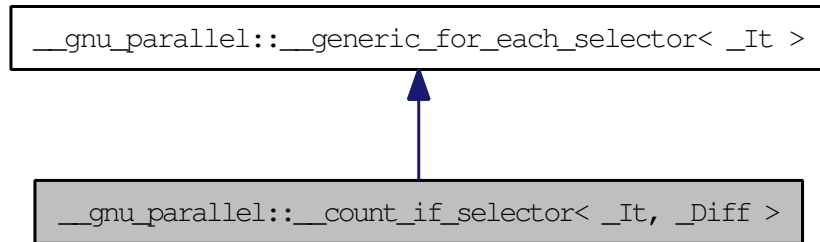
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.96 `__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`

### 5.96.1 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`.

### 5.96.2 Member Function Documentation

**5.96.2.1** `template<typename _It, typename _Diff > template<typename _Op > _Diff \_\_gnu\_parallel::\_\_count\_if\_selector<\_It, \_Diff >::operator\(\) \(\_Op & \_\_o, \_It \_\_i\) \[inline\]`

Functor execution.



## 5.96 `__gnu_parallel::__count_if_selector<_It, _Diff>` Struct Template Reference

1137

### 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`.

## 5.96.3 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()`).

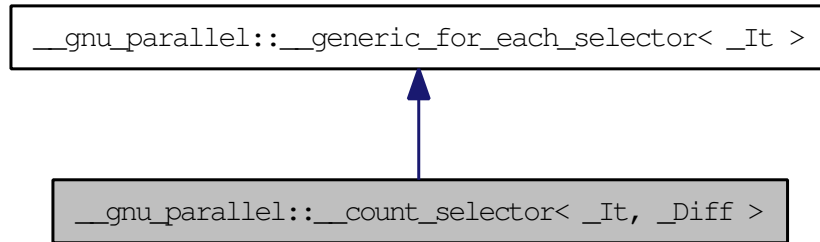
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.97 `__gnu_parallel::__count_selector< _It, _Diff >` Struct Template Reference

`std::count()` selector. 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`

#### 5.97.1 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`.

#### 5.97.2 Member Function Documentation

**5.97.2.1** `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (_ValueType & __v, _It __i) [inline]`

Functor execution.

## 5.97 \_\_gnu\_parallel::\_\_count\_selector<\_It, \_Diff> Struct Template Reference

### 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`.

## 5.97.3 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()`).

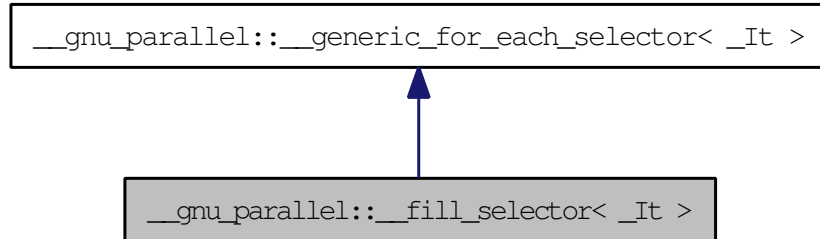
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.98 `__gnu_parallel::__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`

### 5.98.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector< _It >`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

### 5.98.2 Member Function Documentation

**5.98.2.1** `template<typename _It > template<typename _ValueType > bool __gnu_parallel::__fill_selector< _It >::operator() (_ValueType & __v, _It __i) [inline]`

Functor execution.

#### Parameters:

- `__v` Current value.

`__i` iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

### 5.98.3 Member Data Documentation

#### 5.98.3.1 `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()`).

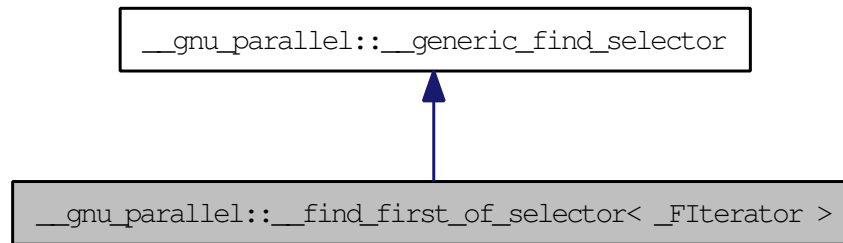
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.99 `__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`

#### 5.99.1 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`.

## 5.99.2 Member Function Documentation

**5.99.2.1** `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`.

**5.99.2.2** `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.

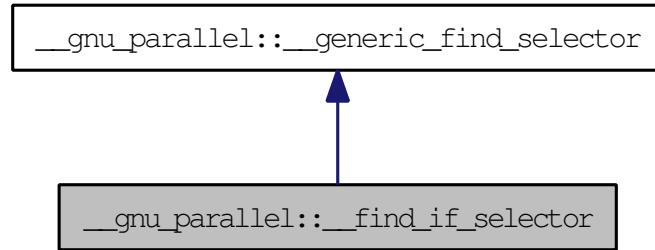
Definition at line 169 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.100 `__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)`

#### 5.100.1 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`.

#### 5.100.2 Member Function Documentation

**5.100.2.1** `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`.

```
5.100.2.2 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.

`__i2` Iterator on second sequence (unused).

`__pred` Find predicate.

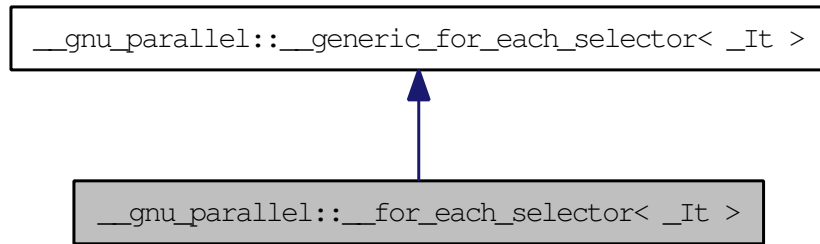
Definition at line 60 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.101 `__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`

### 5.101.1 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`.

### 5.101.2 Member Function Documentation

**5.101.2.1** `template<typename _It> template<typename _Op > bool`  
`__gnu_parallel::__for_each_selector< _It >::operator() (_Op &__o,`  
`__It __i) [inline]`

Functor execution.

## **5.101 \_\_gnu\_parallel::\_\_for\_each\_selector<\_It> Struct Template Reference 147**

### **Parameters:**

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

### **5.101.3 Member Data Documentation**

#### **5.101.3.1 `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()`).

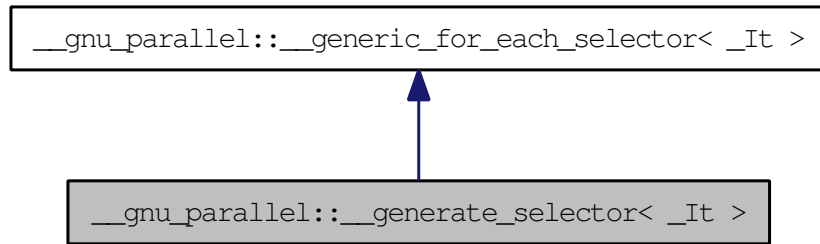
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.102 `__gnu_parallel::__generate_selector< _It >` Struct Template Reference

`std::generate()` selector. 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`

### 5.102.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector< _It >`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

### 5.102.2 Member Function Documentation

**5.102.2.1** `template<typename _It> template<typename _Op > bool`  
`__gnu_parallel::__generate_selector< _It >::operator() (_Op &__o,`  
`__It __i) [inline]`

Functor execution.

## 5.102 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference 149

### Parameters:

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

### 5.102.3 Member Data Documentation

#### 5.102.3.1 `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()`).

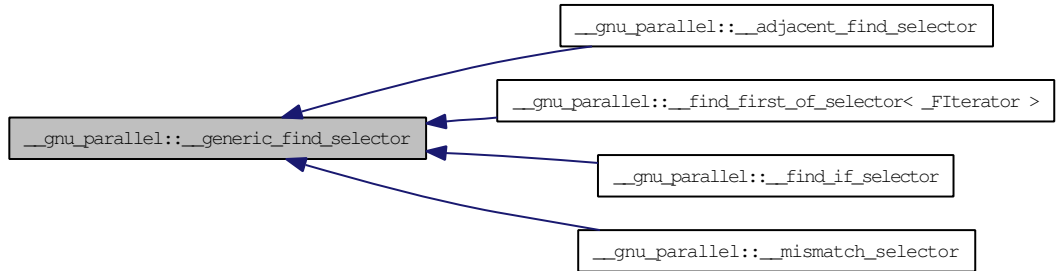
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.103 `__gnu_parallel::__generic_find_selector` Struct Reference

Base class of all `__gnu_parallel::__find_template` selectors. Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



#### 5.103.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

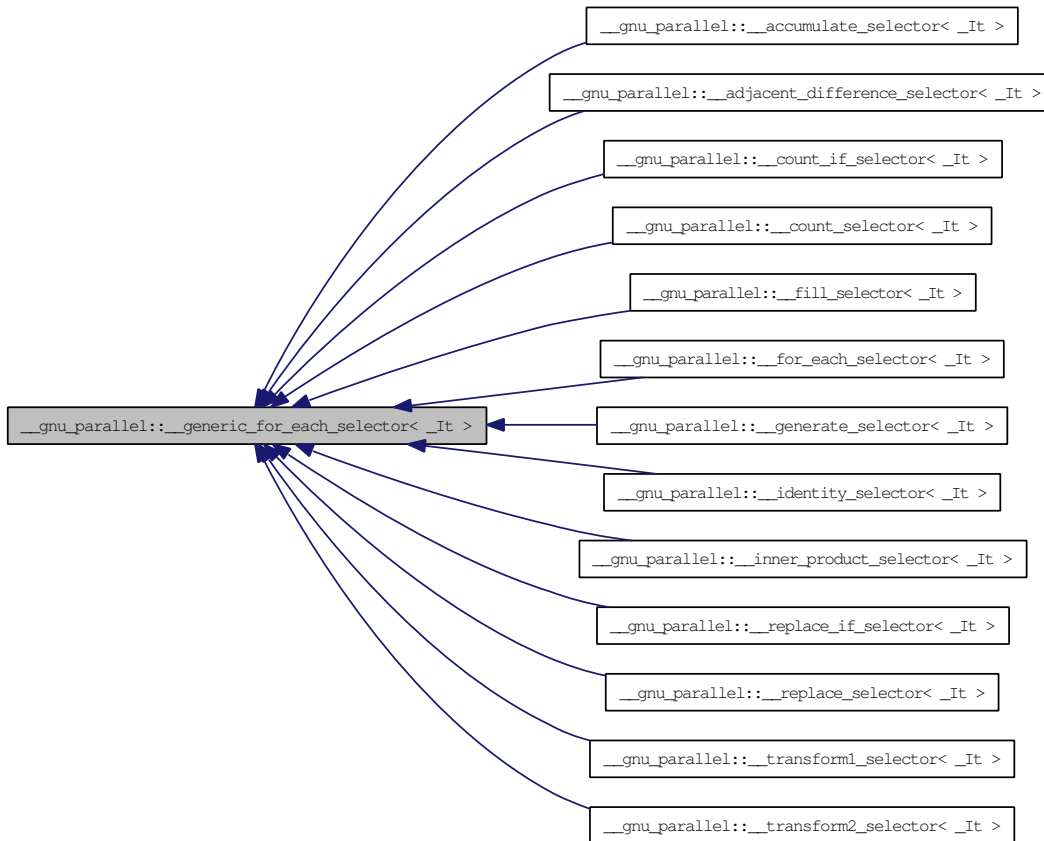
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.104 `__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](#)

### 5.104.1 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`.

### 5.104.2 Member Data Documentation

**5.104.2.1 `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`.

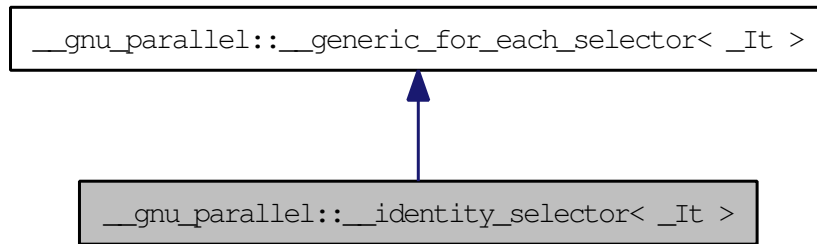
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)



## 5.105 `__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`

#### 5.105.1 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`.

#### 5.105.2 Member Function Documentation

5.105.2.1 `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`.

### 5.105.3 Member Data Documentation

#### 5.105.3.1 `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()`).

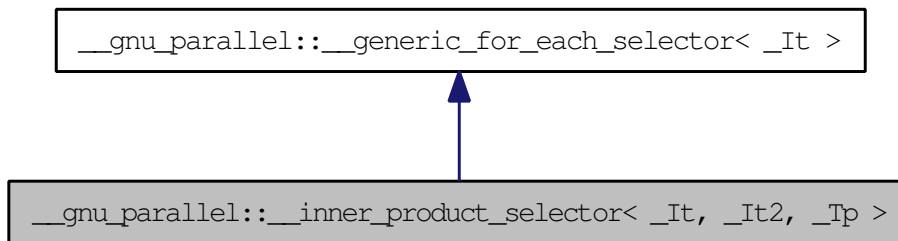
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.106 `__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`

#### 5.106.1 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`.

## 5.106.2 Constructor & Destructor Documentation

**5.106.2.1** `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`.

## 5.106.3 Member Function Documentation

**5.106.3.1** `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.

## 5.106.4 Member Data Documentation

**5.106.4.1** `template<typename _It , typename _It2 , typename _Tp >  
 _It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp  
 >::__begin1_iterator`

Begin iterator of first sequence.

## 5.106 `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>` Struct Template Reference 1157

---

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

### 5.106.4.2 `template<typename _It, typename _It2, typename _Tp> __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()`.

### 5.106.4.3 `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](#)

## 5.107 `__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`

#### 5.107.1 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](#)

## 5.108 `__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`

#### 5.108.1 Detailed Description

`template<typename _Compare, typename _It> struct __gnu_parallel::__min_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

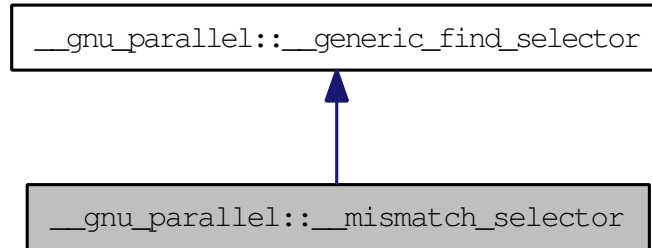
Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.109 `__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)`

### 5.109.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

### 5.109.2 Member Function Documentation

**5.109.2.1** `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`.

**5.109.2.2** `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](#)

## 5.110 `__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.

### Public Member Functions

- `_RAIter3 operator()` (`_RAIterIterator __seqs_begin`, `_RAIterIterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

#### 5.110.1 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](#)

5.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch`< true, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` > Struct Template

Reference 1163  
**5.111 `__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`)

### 5.111.1 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:

- [multiway\\_merge.h](#)

## 5.112 `__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`)

#### 5.112.1 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](#)

5.113 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch`< true, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` > Struct Template

Reference  
5.113 `__gnu_parallel::__multiway_merge_4_variant_`<sup>1165</sup>  
`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`)

### 5.113.1 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](#)

## 5.114 `__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`)

### 5.114.1 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](#)

5.115 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch`< `false`,  
`__stable`, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` > Struct  
Template Reference 1167

5.115 `__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`)

### 5.115.1 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.

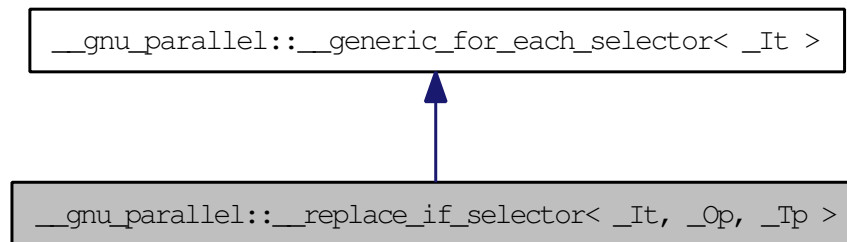
Definition at line 868 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.116 `__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`

#### 5.116.1 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`.



## 5.116.2 Constructor & Destructor Documentation

**5.116.2.1** `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`.

## 5.116.3 Member Function Documentation

**5.116.3.1** `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`.

## 5.116.4 Member Data Documentation

**5.116.4.1** `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()`.

**5.116.4.2** `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`.

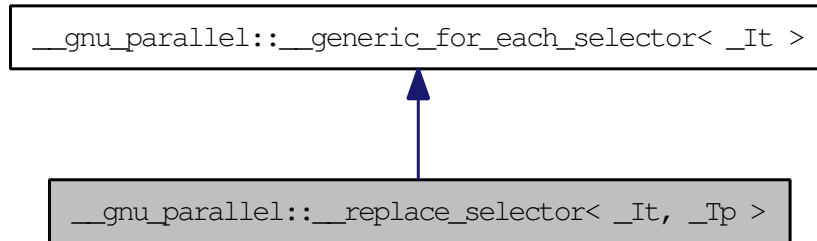
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.117 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

### 5.117 `__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`

#### 5.117.1 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`.

#### 5.117.2 Constructor & Destructor Documentation

5.117.2.1 `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`.

**5.117.3 Member Function Documentation**

**5.117.3.1** `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`.

**5.117.4 Member Data Documentation**

**5.117.4.1** `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()()`.

**5.117.4.2** `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`.

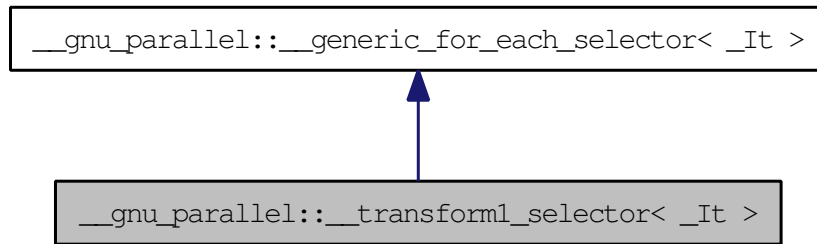
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.118 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

### 5.118 `__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`

#### 5.118.1 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`.

#### 5.118.2 Member Function Documentation

**5.118.2.1** `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`.

### 5.118.3 Member Data Documentation

#### 5.118.3.1 `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`.

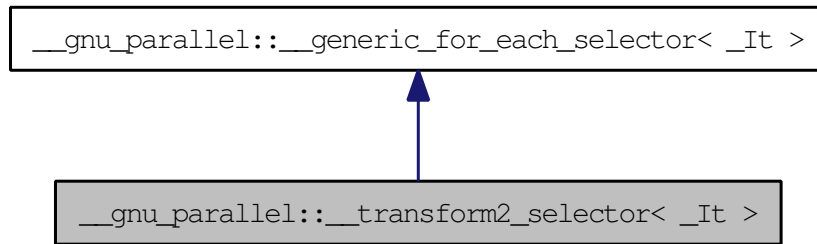
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.119 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

### 5.119 `__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`

#### 5.119.1 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`.

#### 5.119.2 Member Function Documentation

**5.119.2.1** `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`.

### 5.119.3 Member Data Documentation

#### 5.119.3.1 `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`.

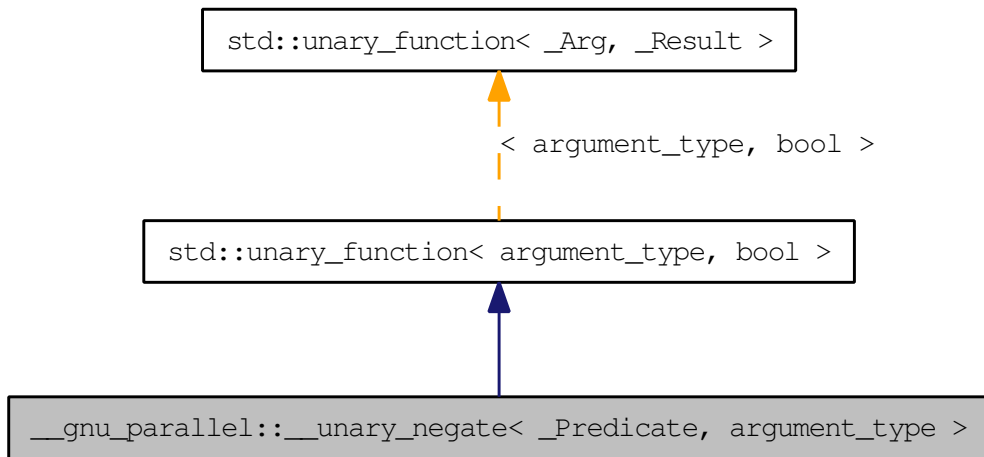
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)



## 5.120 `__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`

### 5.120.1 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`.

### 5.120.2 Member Typedef Documentation

**5.120.2.1** `typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.120.2.2** `typedef bool std::unary_function< argument_type , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.121 `__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`

#### 5.121.1 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`.

## 5.121.2 Constructor & Destructor Documentation

**5.121.2.1** `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData (_RAIter & __source)`  
**[inline]**

Constructor.

Definition at line 83 of file random\_shuffle.h.

## 5.121.3 Member Data Documentation

**5.121.3.1** `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()`.

**5.121.3.2** `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()`.

**5.121.3.3** `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()`.

**5.121 \_\_gnu\_parallel::\_DRandomShufflingGlobalData<\_RAIter> Struct**  
**Template Reference** **1181**

---

**5.121.3.4** `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bits`

Number of bits needed to address the bins.

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()`.

**5.121.3.5** `template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

**5.121.3.6** `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()`.

**5.121.3.7** `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](#)

## 5.122 `__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`

### 5.122.1 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`.

### 5.122.2 Member Data Documentation

**5.122.2.1** `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()`.

**5.122.2.2** `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_bins_begin`

Begin index for bins taken care of by this thread.

**5.122 \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator >  
Struct Template Reference 1183**

---

Definition at line 97 of file random\_shuffle.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs().

**5.122.2.3 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().

**5.122.2.4 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().

**5.122.2.5 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](#)

## 5.123 `__gnu_parallel::_DummyReduct` Struct Reference

Reduction function doing nothing.

### Public Member Functions

- `bool operator()` (bool, bool) const

#### 5.123.1 Detailed Description

Reduction function doing nothing.

Definition at line 298 of file `for_each_selectors.h`.

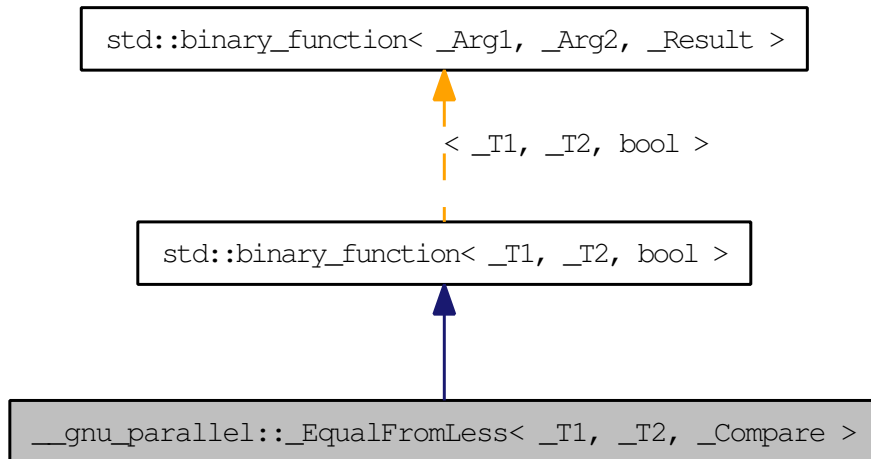
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)



## 5.124 `__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`)

#### 5.124.1 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`.

## 5.124.2 Member Typedef Documentation

**5.124.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.124.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.124.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool  
>::second_argument_type [inherited]`

the type of the second argument

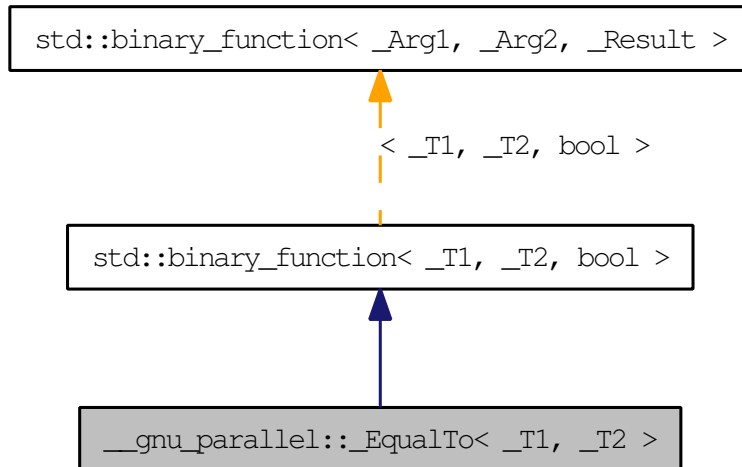
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.125 `__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`

#### 5.125.1 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`.

## 5.125.2 Member Typedef Documentation

**5.125.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.125.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.125.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.126 `__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`)

### 5.126.1 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`.

### 5.126.2 Constructor & Destructor Documentation

**5.126.2.1** `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`.

**5.126.3 Member Function Documentation**

**5.126.3.1** `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`.

**5.126.3.2** `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`.

**5.126.3.3** `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`.

### 5.126.4 Friends And Related Function Documentation

**5.126.4.1** `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`.

**5.126.4.2** `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.

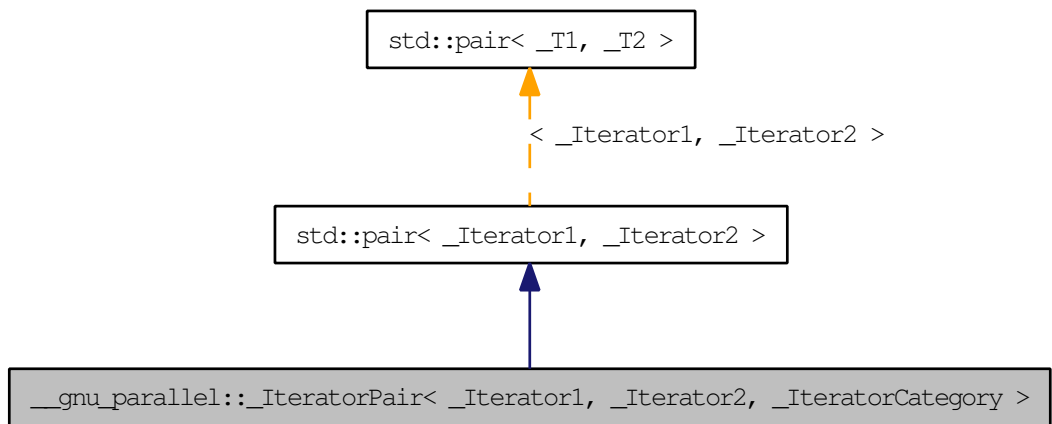
Definition at line 128 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

## 5.127 `__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`

### 5.127.1 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`.

### 5.127.2 Member Typedef Documentation

**5.127.2.1** `typedef _Iterator1 std::pair< _Iterator1 , _Iterator2 >::first_type`  
[`inherited`]

`first_type` is the first bound type

Definition at line 73 of file `stl_pair.h`.

**5.127.2.2** `typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type`  
[`inherited`]

`second_type` is the second bound type

Definition at line 74 of file `stl_pair.h`.

### 5.127.3 Member Data Documentation

**5.127.3.1** `_Iterator1 std::pair< _Iterator1 , _Iterator2 >::first` [`inherited`]

`first` is a copy of the first object

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

### 5.127.3.2 `_Iterator2 std::pair<_Iterator1 ,_Iterator2 >::second` [`inherited`]

`second` is a copy of the second object

Definition at line 77 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.128 `__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
- const `_IteratorTriple` **operator--** (int)
- `_IteratorTriple` & **operator--** ()
- `_IteratorTriple` & **operator=** (const `_IteratorTriple` &\_\_other)

### Public Attributes

- `_Iterator1` **\_M\_first**
- `_Iterator2` **\_M\_second**
- `_Iterator3` **\_M\_third**

#### 5.128.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _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](#)

## 5.129 `__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`

#### 5.129.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Job<_DifferenceTp>`

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

#### 5.129.2 Member Data Documentation

**5.129.2.1** `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()`.

**5.129.2.2** `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()`.

**5.129.2.3** `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`.

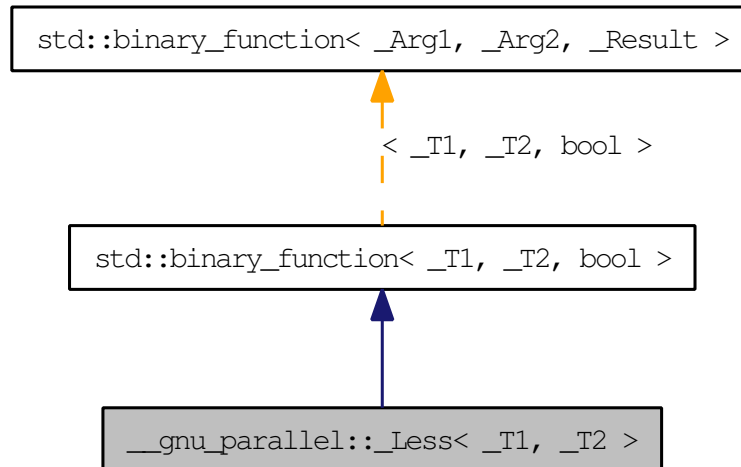
Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

## 5.130 `__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 _T2 &__t2, const _T1 &__t1) const`
- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

#### 5.130.1 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`.

## 5.130.2 Member Typedef Documentation

**5.130.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.130.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.130.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

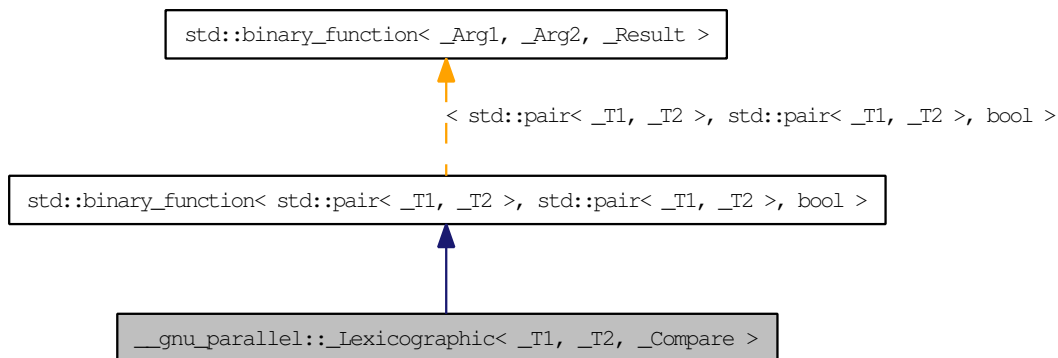
The documentation for this struct was generated from the following file:

- [parallel/base.h](#)



## 5.131 `__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`

#### 5.131.1 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`.

## 5.131.2 Member Typedef Documentation

**5.131.2.1** `typedef std::pair<_T1,_T2> std::binary_function< std::pair<_T1,_T2>, std::pair<_T1,_T2>, bool >::first_argument_type`  
[`inherited`]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.131.2.2** `typedef bool std::binary_function< std::pair<_T1,_T2>, std::pair<_T1,_T2>, bool >::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.131.2.3** `typedef std::pair<_T1,_T2> std::binary_function< std::pair<_T1,_T2>, std::pair<_T1,_T2>, bool >::second_argument_type`  
[`inherited`]

the type of the second argument

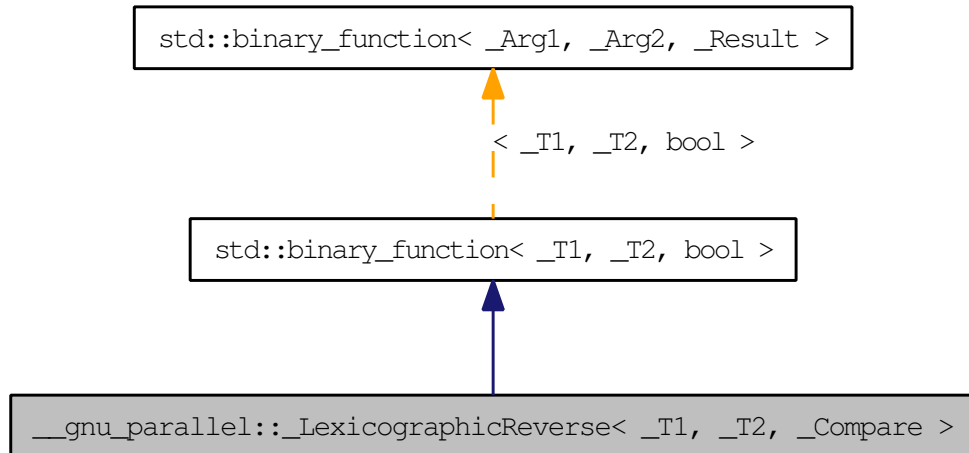
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 5.132 `__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`

#### 5.132.1 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`.

## 5.132.2 Member Typedef Documentation

**5.132.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.132.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.132.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool  
>::second_argument_type [inherited]`

the type of the second argument

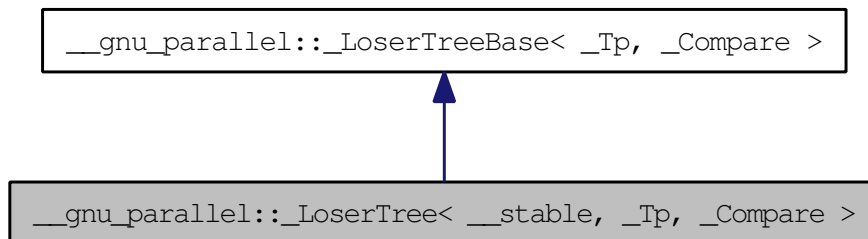
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 5.133 `__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`

#### 5.133.1 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.

## 5.133.2 Member Function Documentation

**5.133.2.1** `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.

References `std::swap()`.

**5.133.2.2** `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`.

**5.133.2.3** `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`,

## 5.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference 1207

---

`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.133.3 Member Data Documentation

#### 5.133.3.1 `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`.

#### 5.133.3.2 `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()`.

#### 5.133.3.3 `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()`.

#### 5.133.3.4 `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()`.

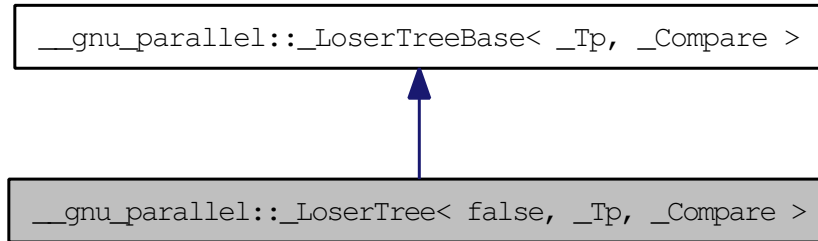
The documentation for this class was generated from the following file:

- [losertree.h](#)



## 5.134 `__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`

#### 5.134.1 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 255 of file losertree.h.

## 5.134.2 Member Function Documentation

**5.134.2.1** `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 317 of file losertree.h.

References `std::swap()`.

**5.134.2.2** `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`.

**5.134.2.3** `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 277 of file losertree.h.

## 5.134 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference 1211

---

**5.134.2.4** `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`.

## 5.134.3 Member Data Documentation

**5.134.3.1** `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`.

**5.134.3.2** `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()`.

**5.134.3.3** `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()`.

**5.134.3.4** `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`.

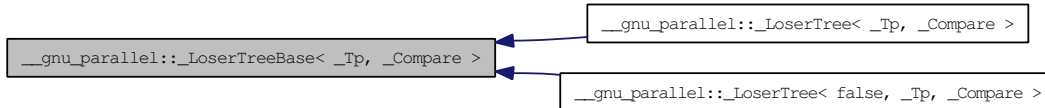
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.135 `__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`

#### 5.135.1 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`.

## 5.135.2 Constructor & Destructor Documentation

**5.135.2.1** `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`.

**5.135.2.2** `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`.

### 5.135.3 Member Function Documentation

**5.135.3.1** `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`.

**5.135.3.2** `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`, `__gnu_parallel::LoserTreeBase<_Tp, _Compare >::__M_losers`, `__gnu_parallel::LoserTreeBase<_Tp, _Compare >::__Loser::__M_source`, and `__gnu_parallel::LoserTreeBase<_Tp, _Compare >::__Loser::__M_sup`.

### 5.135.4 Member Data Documentation

**5.135.4.1** `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`.

**5.135.4.2** `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()`.

**5.135.4.3** `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()`.

**5.135.4.4** `template<typename _Tp , typename _Compare > _Loser*`  
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`  
`[protected]`

[\\_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](#)



## 5.136 `__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](#)

#### 5.136.1 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`.

#### 5.136.2 Member Data Documentation

**5.136.2.1** `template<typename _Tp, typename _Compare> __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()`.

**5.136.2.2** `template<typename _Tp, typename _Compare> __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()`.

**5.136.2.3** `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.

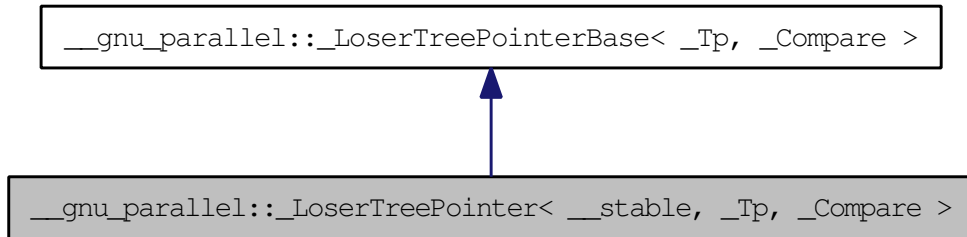
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

## 5.137 `__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`

#### 5.137.1 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.

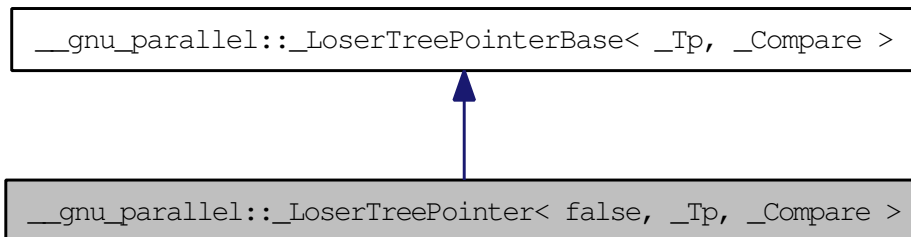
Definition at line 401 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.138 `__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`

#### 5.138.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation. The stable variant is above.

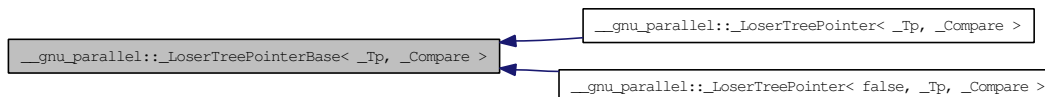
Definition at line 482 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.139 `__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`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.139.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`

Base class of `_Loser` Tree implementation using pointers.

Definition at line 349 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)



## 5.140 `__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`

### 5.140.1 Detailed Description

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare >::_Loser`

Internal representation of [\\_LoserTree](#) \_\_elements.

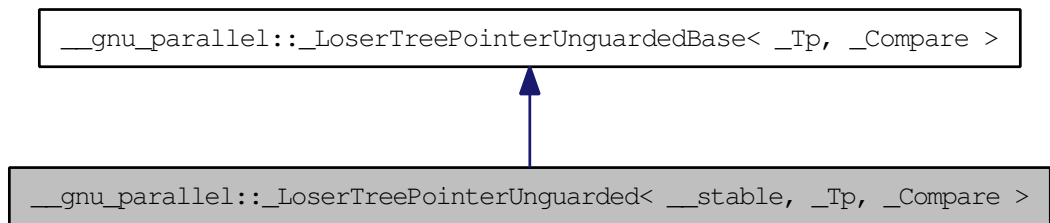
Definition at line 353 of file `losertree.h`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

## 5.141 `__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`

#### 5.141.1 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.

**5.141 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare`**  
**> Class Template Reference** **1227**

---

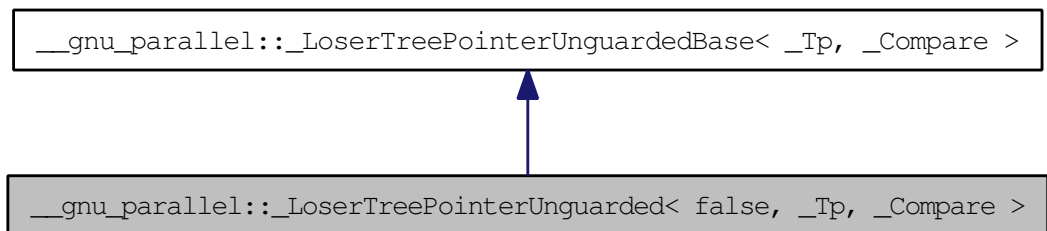
Definition at line 868 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.142 `__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**

#### 5.142.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded<false, _Tp, _Compare >
```

Unstable unguarded [\\_LoserTree](#) variant storing pointers. Stable variant is above.

**5.142 \_\_gnu\_parallel::\_LoserTreePointerUnguarded< false, \_Tp, \_Compare >**  
**Class Template Reference** **1229**

---

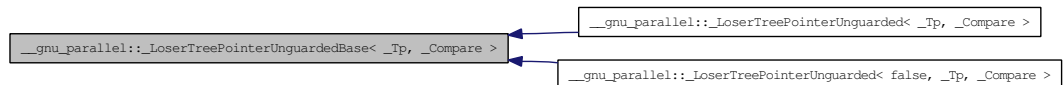
Definition at line 953 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.143 `__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)

### Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

#### 5.143.1 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 805 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.144 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference 1231

### 5.144 `__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`

#### 5.144.1 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`.

#### 5.144.2 Member Data Documentation

5.144.2.1 `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:

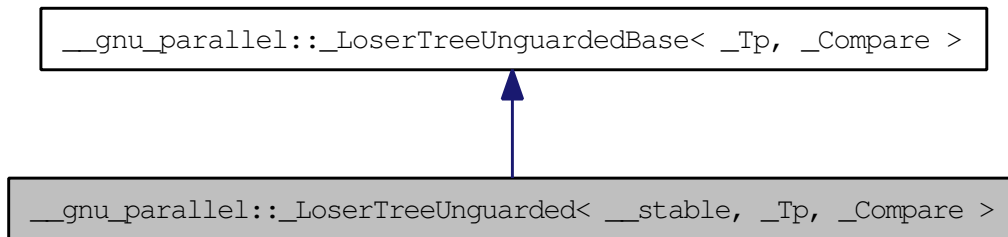
- [multiway\\_merge.h](#)



5.145 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class  
Template Reference 1233

## 5.145 `__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`

#### 5.145.1 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.

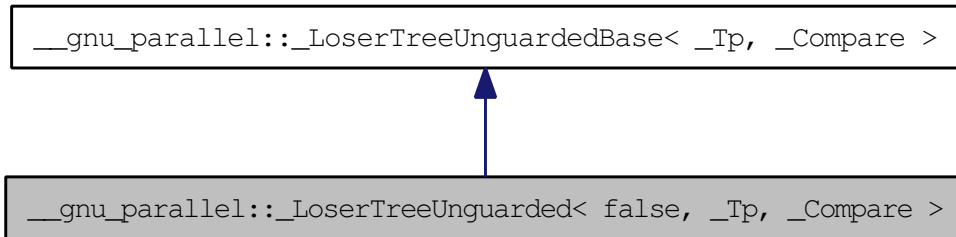
Definition at line 627 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.146 `__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`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.146.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
```

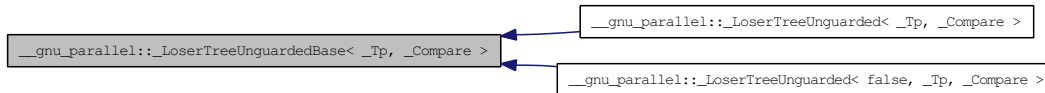
Non-Stable implementation of unguarded `_LoserTree`. Stable implementation is above.  
Definition at line 713 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.147 `__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`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.147.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`

Base class for unguarded `_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 564 of file `losertree.h`.

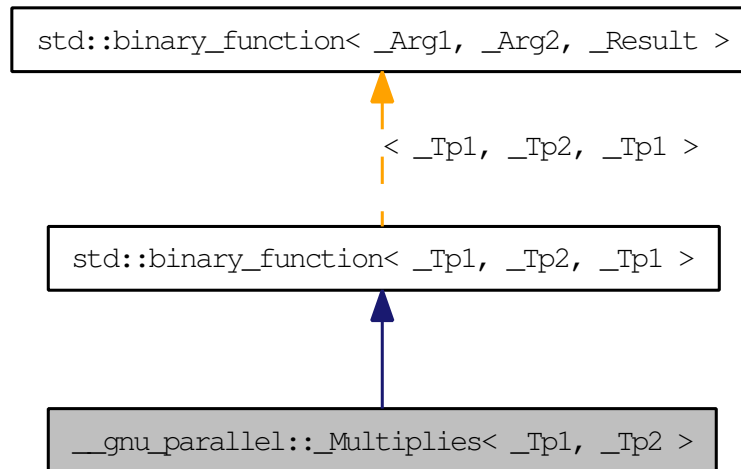
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.148 `__gnu_parallel::_Multiplies<_Tp1, _Tp2 >` Struct Template Reference 239

### 5.148 `__gnu_parallel::_Multiplies<_Tp1, _Tp2 >` Struct Template Reference

Similar to `std::multiplies`, but allows two different types. Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2 >`:



#### Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Tp1` `result_type`
- typedef `_Tp2` `second_argument_type`
- typedef `__typeof__(*)(const _Tp1 &__x, const _Tp2 &__y static_cast<_Tp2 * >(NULL)) __result; __resultoperator(`

#### 5.148.1 Detailed Description

```
template<typename _Tp1, typename _Tp2> struct __gnu_parallel::_Multiplies<_Tp1, _Tp2 >
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 300 of file `parallel/base.h`.

## 5.148.2 Member Typedef Documentation

**5.148.2.1** `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Tp1 >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.148.2.2** `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Tp1 >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.148.2.3** `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Tp1 >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)



## 5.149 `__gnu_parallel::_Nothing` Struct Reference

Functor doing nothing.

### Public Member Functions

- `template<typename _It >`  
`void operator() (_It)`

#### 5.149.1 Detailed Description

Functor doing nothing. For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

#### 5.149.2 Member Function Documentation

##### 5.149.2.1 `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](#)

## 5.150 `__gnu_parallel::_Piece<_DifferenceTp>` Struct Template Reference

Subsequence description.

### Public Types

- `typedef _DifferenceTp _DifferenceType`

### Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

#### 5.150.1 Detailed Description

```
template<typename _DifferenceTp> struct __gnu_parallel::_Piece< _-
DifferenceTp >
```

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

#### 5.150.2 Member Data Documentation

**5.150.2.1** `template<typename _DifferenceTp > _DifferenceType  
__gnu_parallel::_Piece<_DifferenceTp >::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

**5.150.2.2** `template<typename _DifferenceTp > _DifferenceType  
__gnu_parallel::_Piece<_DifferenceTp >::_M_end`

End of subsequence.

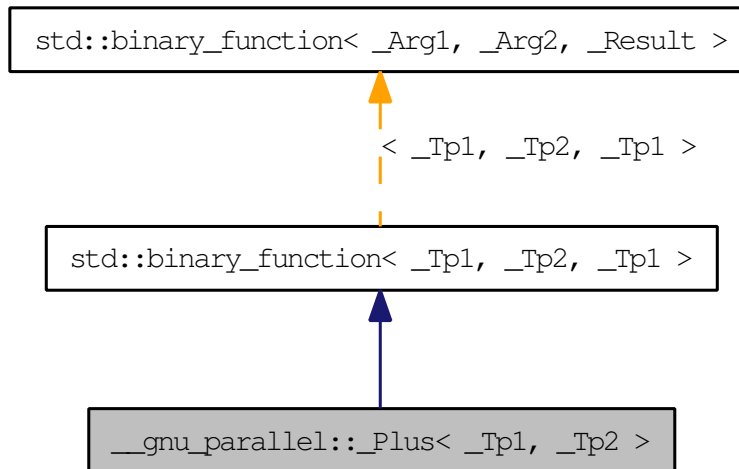
Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.151 `__gnu_parallel::_Plus< _Tp1, _Tp2 >` Struct Template Reference

Similar to `std::plus`, but allows two different types. Inheritance diagram for `__gnu_parallel::_Plus< _Tp1, _Tp2 >`:



### Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Tp1` `result_type`
- typedef `_Tp2` `second_argument_type`
- typedef `__typeof__(*)(const _Tp1 &__x, const _Tp2 &__y static_cast<_Tp2 * >(NULL))+static_cast<_Tp2 * >(NULL)) __result; __resultoperator(`

#### 5.151.1 Detailed Description

```
template<typename _Tp1, typename _Tp2> struct __gnu_parallel::_Plus< _Tp1, _Tp2 >
```

Similar to `std::plus`, but allows two different types.

Definition at line 275 of file `parallel/base.h`.

## 5.151.2 Member Typedef Documentation

**5.151.2.1** `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Tp1  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.151.2.2** `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Tp1  
>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.151.2.3** `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Tp1  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.152 `__gnu_parallel::_PMWMSortingData<_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`

### 5.152.1 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`.

### 5.152.2 Member Data Documentation

**5.152.2.1** `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()`.

### 5.152.2.2 `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()`.

### 5.152.2.3 `template<typename _RAIter> std::vector<_Piece<_- _DifferenceType> >* __gnu_parallel::PMWMSortingData< _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()`.

### 5.152.2.4 `template<typename _RAIter> _ValueType* __gnu_parallel::PMWMSortingData< _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()`.

### 5.152.2.5 `template<typename _RAIter> _RAIter __gnu_parallel::PMWMSortingData< _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()`.

### 5.152.2.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSortingData< _RAIter >::_M_starts`

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

## 5.152 \_\_gnu\_parallel::\_PMWMSortingData<\_RAIter> Struct Template Reference 1247

---

Referenced by `__gnu_parallel::_determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

### 5.152.2.7 `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](#)

## 5.153 `__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

#### 5.153.1 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.
- `_DifferenceType` Sequence difference type.

Definition at line 386 of file `parallel/base.h`.

#### 5.153.2 Constructor & Destructor Documentation

```
5.153.2.1 template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp
>::_PseudoSequence (const _Tp & __val, _DifferenceType __count)
[inline]
```

Constructor.



## 5.153 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference 1249

---

### Parameters:

- `_M_val` Element of the sequence.
- `__count` Number of (virtual) copies.

Definition at line 398 of file `parallel/base.h`.

### 5.153.3 Member Function Documentation

**5.153.3.1** `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin ()  
const [inline]`

Begin iterator.

Definition at line 403 of file `parallel/base.h`.

**5.153.3.2** `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end ()  
const [inline]`

End iterator.

Definition at line 408 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.154 `__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)
- `_DifferenceType operator!=` (const `_PseudoSequenceIterator` &\_\_i2)
- const `_Tp` & `operator*` () const
- const `_PseudoSequenceIterator` `operator++` (int)
- `_PseudoSequenceIterator` & `operator++` ()
- `_DifferenceType operator-` (const `_PseudoSequenceIterator` &\_\_i2)
- bool `operator==` (const `_PseudoSequenceIterator` &\_\_i2)
- const `_Tp` & `operator[]` (`_DifferenceType`) const

#### 5.154.1 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.
- `_DifferenceType` Sequence difference type.

Definition at line 332 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.155 `__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`

#### 5.155.1 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`.

#### 5.155.2 Member Typedef Documentation

**5.155.2.1** `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`.

### 5.155.3 Constructor & Destructor Documentation

#### 5.155.3.1 `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`.

### 5.155.4 Member Data Documentation

#### 5.155.4.1 `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()`.

#### 5.155.4.2 `template<typename _RAIter> _Piece __gnu_parallel::_QSBThreadLocal<_RAIter >::_M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

#### 5.155.4.3 `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()`.

**5.155 \_\_gnu\_parallel::\_QSBThreadLocal<\_RAIter> Struct Template Reference**

1253

**5.155.4.4 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().

**5.155.4.5 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](#)

## 5.156 `__gnu_parallel::_RandomNumber` Class Reference

Random number generator, based on the Mersenne twister.

### Public Member Functions

- `_RandomNumber` (`uint32_t __seed`, `uint64_t _M_supremum=0x100000000ULL`)
- `_RandomNumber` ()
- `unsigned long __genrand_bits` (`int __bits`)
- `uint32_t operator()` (`uint64_t local_supremum`)
- `uint32_t operator()` ()

#### 5.156.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

#### 5.156.2 Constructor & Destructor Documentation

##### 5.156.2.1 `__gnu_parallel::_RandomNumber::_RandomNumber` () [`inline`]

Default constructor. Seed with 0.

Definition at line 74 of file `random_number.h`.

##### 5.156.2.2 `__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`.

### 5.156.3 Member Function Documentation

#### 5.156.3.1 `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`.

#### 5.156.3.2 `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`.

#### 5.156.3.3 `uint32_t __gnu_parallel::_RandomNumber::operator() ()` [`inline`]

Generate unsigned random 32-bit integer.

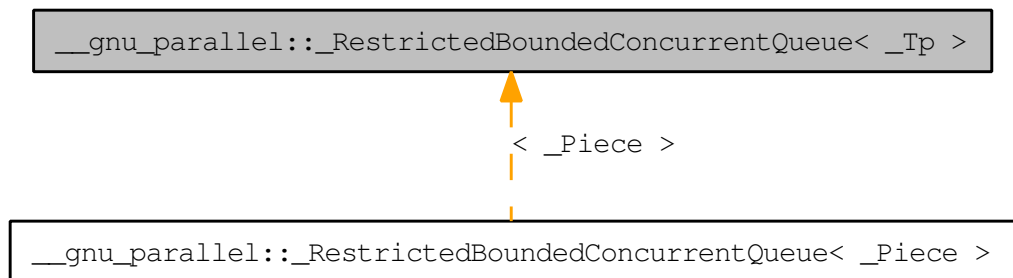
Definition at line 94 of file `random_number.h`.

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

## 5.157 `__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`)

#### 5.157.1 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`.



## 5.157.2 Constructor & Destructor Documentation

**5.157.2.1** `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`.

**5.157.2.2** `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`.

## 5.157.3 Member Function Documentation

**5.157.3.1** `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`.

**5.157.3.2** `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()`.

**5.157.3.3** `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](#)

5.158 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference 1259

---

## 5.158 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Stable sorting functor.

### Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

#### 5.158.1 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](#)

## 5.159 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference`

Non-\_\_stable sorting functor.

### Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

#### 5.159.1 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](#)

## 5.160 `__gnu_parallel::_Settings` Struct Reference

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

### Public Member Functions

- `__attribute__((__const__))` static const `_Settings` &get() throw ()

### Static Public Member Functions

- 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`
- `_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`

### 5.160.1 Detailed Description

class [\\_Settings](#) Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file settings.h.

### 5.160.2 Member Function Documentation

#### 5.160.2.1 `__gnu_parallel::_Settings::__attribute__ ((__const__)) const throw ()`

Get the global settings.

#### 5.160.2.2 `static void __gnu_parallel::_Settings::set (_Settings &) throw ()` [`static`]

Set the global settings.

### 5.160.3 Member Data Documentation

#### 5.160.3.1 `_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n`

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

#### 5.160.3.2 `unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n`

Minimal input size for adjacent\_difference.

Definition at line 142 of file settings.h.

#### 5.160.3.3 `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()`.

#### 5.160.3.4 `_SequenceIndex __gnu_parallel::_Settings::count_minimal_n`

Minimal input size for count and count\_if.

Definition at line 145 of file settings.h.

#### 5.160.3.5 `_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n`

Minimal input size for fill.

Definition at line 148 of file settings.h.

#### 5.160.3.6 `double __gnu_parallel::_Settings::find_increasing_factor`

Block size increase factor for find.

Definition at line 151 of file settings.h.

Referenced by `__gnu_parallel::_find_template()`.

**5.160.3.7** `_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()`.

**5.160.3.8** `_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size`

Maximal block size for find.

Definition at line 157 of file settings.h.

Referenced by `__gnu_parallel::_find_template()`.

**5.160.3.9** `_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()`.

**5.160.3.10** `_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n`

Minimal input size for `for_each`.

Definition at line 163 of file settings.h.

**5.160.3.11** `_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for `generate`.

Definition at line 166 of file settings.h.

**5.160.3.12** `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.



**5.160.3.13** `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()`.

**5.160.3.14** `_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for `max_element`.

Definition at line 169 of file settings.h.

**5.160.3.15** `_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for `merge`.

Definition at line 172 of file settings.h.

**5.160.3.16** `unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for `merge`.

Definition at line 175 of file settings.h.

**5.160.3.17** `_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for `min_element`.

Definition at line 178 of file settings.h.

**5.160.3.18** `int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for `multiway_merge`.

Definition at line 184 of file settings.h.

**5.160.3.19** `_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n`

Minimal input size for `multiway_merge`.

Definition at line 181 of file settings.h.

#### **5.160.3.20 unsigned int \_\_gnu\_parallel::\_Settings::multiway\_merge\_oversampling**

Oversampling factor for multiway\_merge.

Definition at line 187 of file settings.h.

#### **5.160.3.21 \_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().

#### **5.160.3.22 \_SequenceIndex \_\_gnu\_parallel::\_Settings::partial\_sort\_minimal\_n**

Minimal input size for partial\_sort.

Definition at line 203 of file settings.h.

#### **5.160.3.23 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().

#### **5.160.3.24 unsigned int \_\_gnu\_parallel::\_Settings::partial\_sum\_minimal\_n**

Minimal input size for partial\_sum.

Definition at line 210 of file settings.h.

#### **5.160.3.25 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().

**5.160.3.26** `_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()`.

**5.160.3.27** `_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()`.

**5.160.3.28** `_SequenceIndex __gnu_parallel::_Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

**5.160.3.29** `unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n`

Minimal input size for `random_shuffle`.

Definition at line 213 of file settings.h.

**5.160.3.30** `_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Definition at line 216 of file settings.h.

**5.160.3.31** `_SequenceIndex __gnu_parallel::_Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Definition at line 273 of file settings.h.

**5.160.3.32** `_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file settings.h.

**5.160.3.33** `_SequenceIndex __gnu_parallel::_Settings::set_intersection_ - minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

**5.160.3.34** `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_ - difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

**5.160.3.35** `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

**5.160.3.36** `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

**5.160.3.37** `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

**5.160.3.38** `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`.

**5.160.3.39** `_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`.

#### 5.160.3.40 `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()`.

#### 5.160.3.41 `_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for `parallel std::transform`.

Definition at line 244 of file `settings.h`.

#### 5.160.3.42 `_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](#)

## 5.161 `__gnu_parallel::_SplitConsistently`< `__exact`, `_RAIter`, `_Compare`, `_SortingPlacesIterator` > Struct Template Reference

Split consistently.

### 5.161.1 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`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

**5.162** `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference 1271

---

## **5.162** `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Split by sampling.

### **Public Member Functions**

- `void operator() (const \_ThreadIndex __iam, \_PMWMSortingData< \_RAIter > *__sd, \_Compare &__comp, const typename std::iterator\_traits< \_RAIter >::difference_type __num_samples) const`

### **5.162.1 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](#)

## 5.163 `__gnu_parallel::_SplitConsistently`< `true`, `_RAIter`, `_Compare`, `_SortingPlacesIterator` > Struct Template Reference

Split by exact splitting.

### Public Member Functions

- `void operator()` (const `_ThreadIndex __iam`, `_PMWMSortingData`< `_RAIter` > \*`__sd`, `_Compare` &`__comp`, const typename `std::iterator_traits`< `_RAIter` >::`difference_type __num_samples`) const

#### 5.163.1 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](#)



## 5.164 `__gnu_parallel::_VoidFunctor<_ValueType>` Class Template Reference 273

### 5.164 `__gnu_parallel::_VoidFunctor<_ValueType>` Class Template Reference

Functor that does nothing.

#### 5.164.1 Detailed Description

```
template<typename _ValueType> class __gnu_parallel::_VoidFunctor<_ValueType>
>
```

Functor that does nothing.

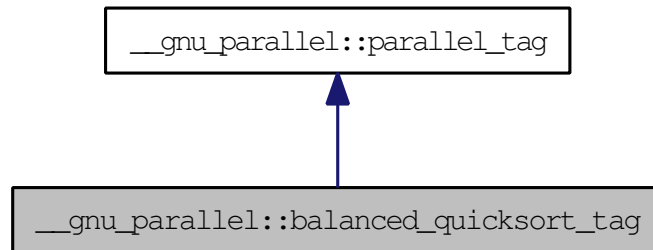
Definition at line 418 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.165 `__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)

#### 5.165.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

#### 5.165.2 Member Function Documentation

##### 5.165.2.1 `_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`().

5.165.2.2 `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.

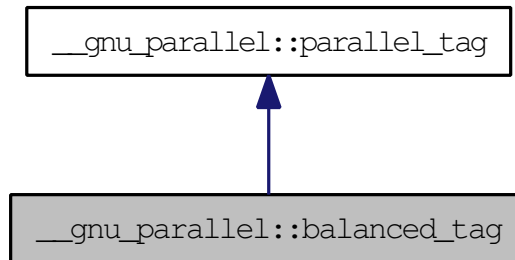
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.166 `__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)`

#### 5.166.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file `tags.h`.

#### 5.166.2 Member Function Documentation

##### 5.166.2.1 `_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()`.

**5.166.2.2** `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.

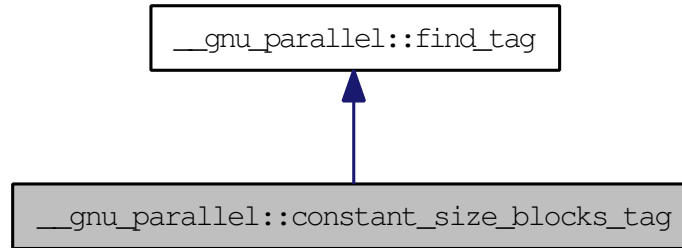
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.167 `__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`:



### 5.167.1 Detailed Description

Selects the constant block size variant for `std::find()`.

**See also:**

[\\_GLIBCXX\\_FIND\\_CONSTANT\\_SIZE\\_BLOCKS](#)

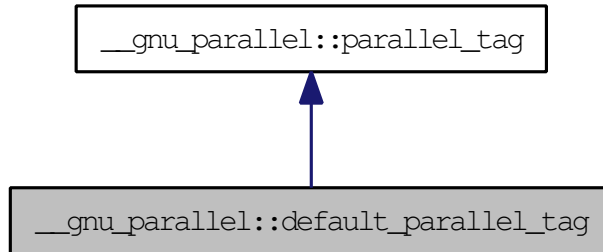
Definition at line 178 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.168 `__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)

#### 5.168.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 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.

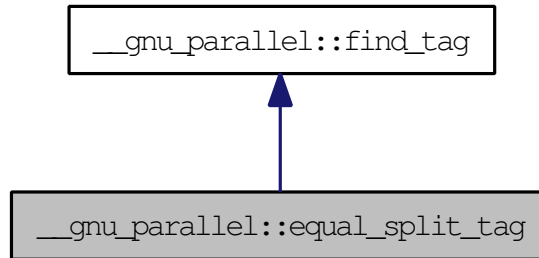
The documentation for this struct was generated from the following file:

- [tags.h](#)



## 5.169 `__gnu_parallel::equal_split_tag` Struct Reference

Selects the equal splitting variant for `std::find()`. Inheritance diagram for `__gnu_parallel::equal_split_tag`:



### 5.169.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

**See also:**

[\\_GLIBCXX\\_FIND\\_EQUAL\\_SPLIT](#)

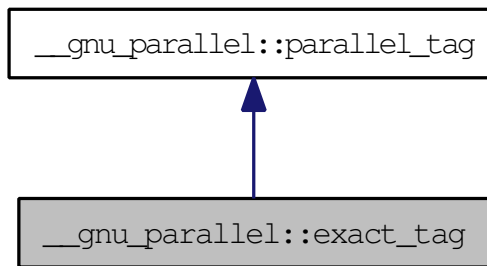
Definition at line 182 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.170 `__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`)

### 5.170.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 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.

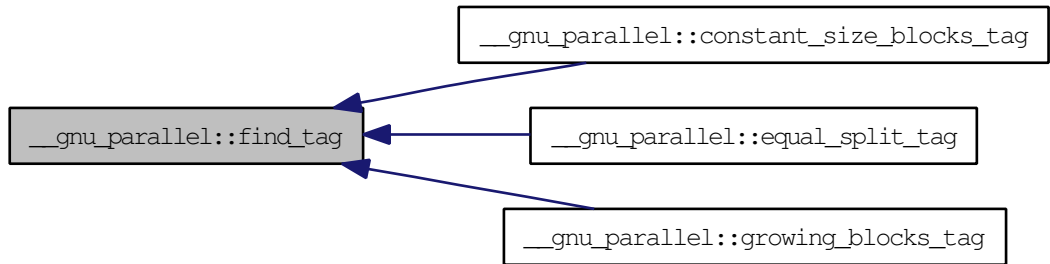
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.171 `__gnu_parallel::find_tag` Struct Reference

Base class for for `std::find()` variants. Inheritance diagram for `__gnu_parallel::find_tag`:



### 5.171.1 Detailed Description

Base class for for `std::find()` variants.

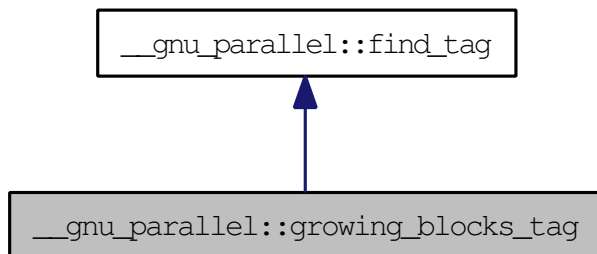
Definition at line 104 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.172 `__gnu_parallel::growing_blocks_tag` Struct Reference

Selects the growing block size variant for `std::find()`. Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



### 5.172.1 Detailed Description

Selects the growing block size variant for `std::find()`.

**See also:**

[\\_GLIBCXX\\_FIND\\_GROWING\\_BLOCKS](#)

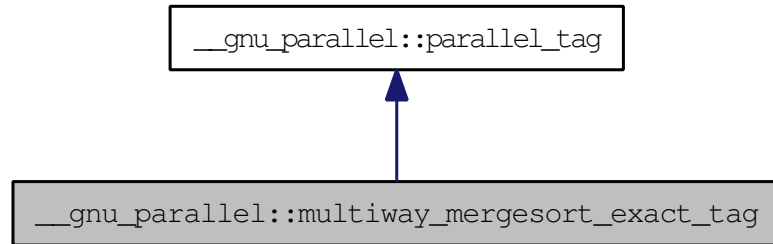
Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.173 `__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`)

#### 5.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.  
Definition at line 137 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 \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag Struct Reference 1287

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.

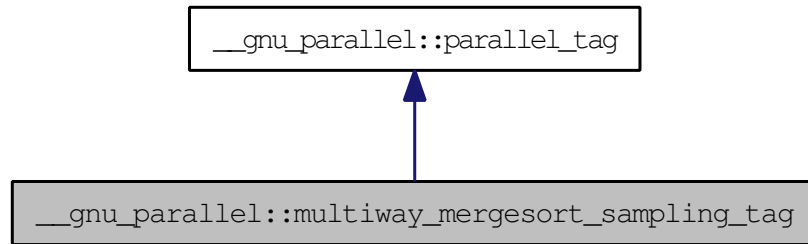
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.174 `__gnu_parallel::multiway_mergesort_samplng_tag` Struct Reference

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time. Inheritance diagram for `__gnu_parallel::multiway_mergesort_samplng_tag`:



### Public Member Functions

- `multiway_mergesort_samplng_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 5.174.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

#### 5.174.2 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 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag Struct Reference 1289

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.

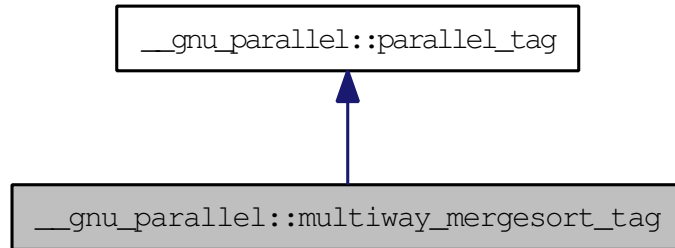
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.175 `__gnu_parallel::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)

#### 5.175.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

#### 5.175.2 Member Function Documentation

##### 5.175.2.1 `_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`().

**5.175.2.2** `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.

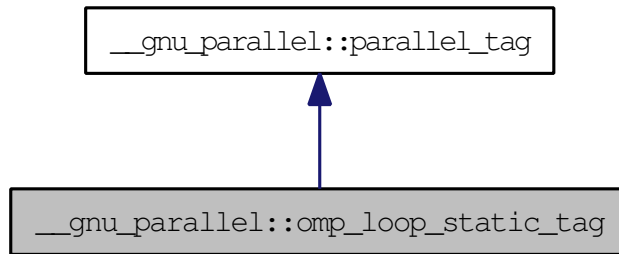
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.176 `__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 \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.176.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.  
Definition at line 100 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.

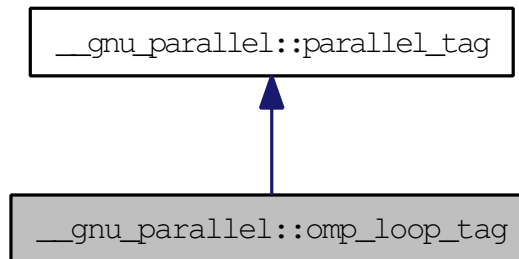
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.177 `__gnu_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\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.177.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

#### 5.177.2 Member Function Documentation

##### 5.177.2.1 `_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()`.

**5.177.2.2** 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.

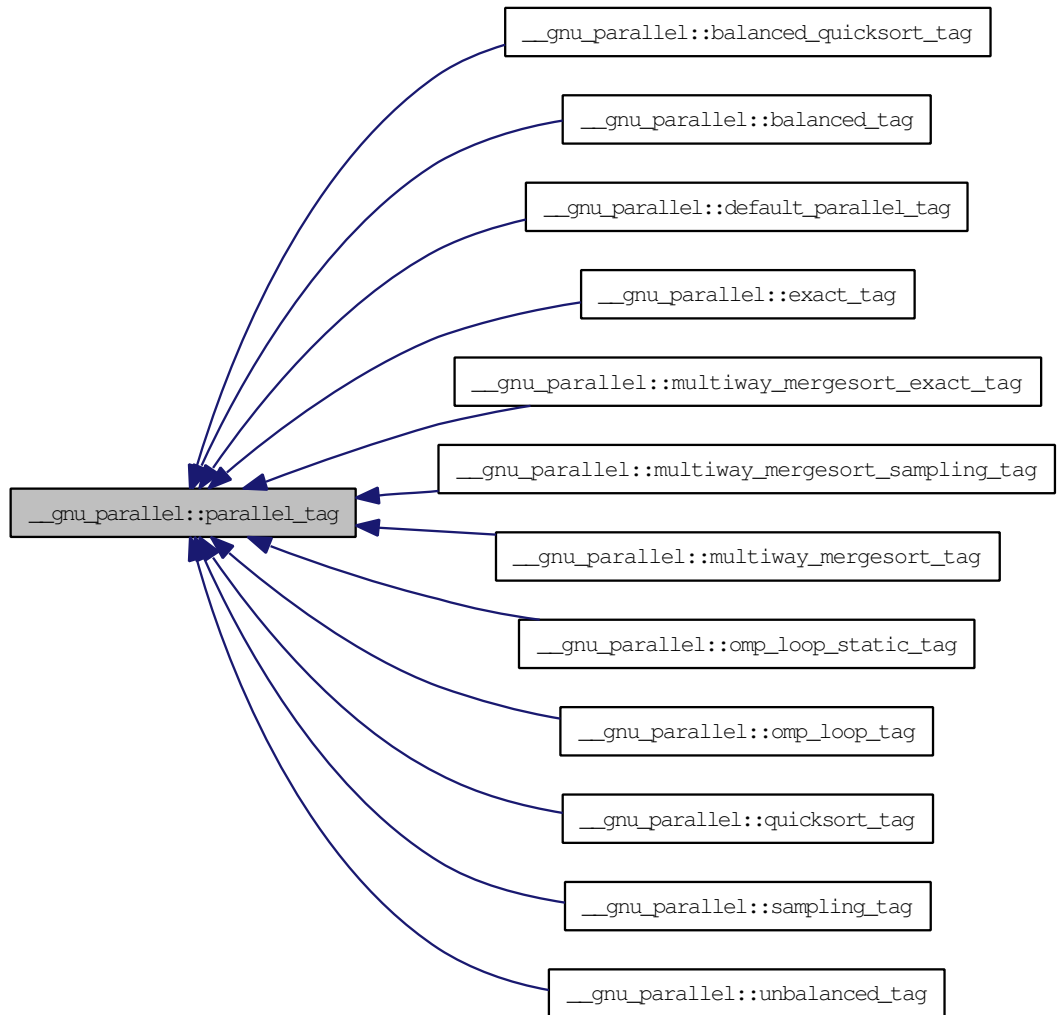
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.178 `__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` (`_ThreadIndex` \_\_num\_threads)
- `parallel_tag` ()
- `_ThreadIndex` \_\_get\_num\_threads ()



- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

### 5.178.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file `tags.h`.

### 5.178.2 Constructor & Destructor Documentation

#### 5.178.2.1 `__gnu_parallel::parallel_tag::parallel_tag ()` [`inline`]

Default constructor. Use default number of threads.

Definition at line 53 of file `tags.h`.

#### 5.178.2.2 `__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`.

### 5.178.3 Member Function Documentation

#### 5.178.3.1 `_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()`.

**5.178.3.2** `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.

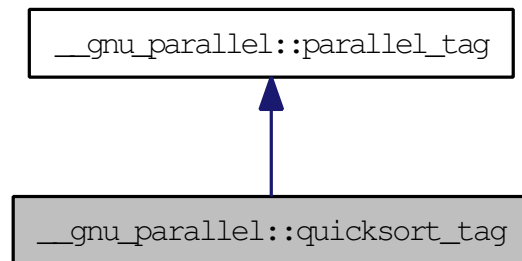
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.179 `__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`)

### 5.179.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file `tags.h`.

### 5.179.2 Member Function Documentation

#### 5.179.2.1 `_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` ().

**5.179.2.2** `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.

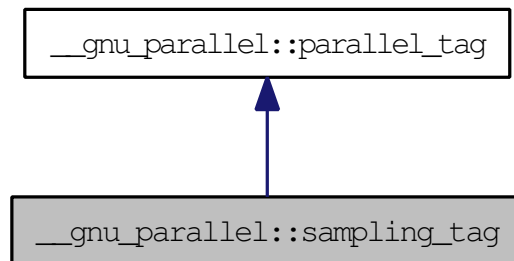
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.180 `__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`)

### 5.180.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file `tags.h`.

### 5.180.2 Member Function Documentation

#### 5.180.2.1 `_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` ().

**5.180.2.2** `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](#)

## 5.181 `__gnu_parallel::sequential_tag` Struct Reference

Forces sequential execution at compile time.

### 5.181.1 Detailed Description

Forces sequential execution at compile time.

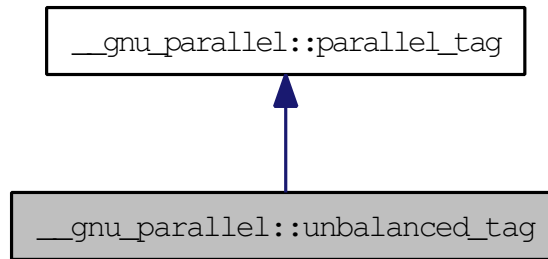
Definition at line 42 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.182 `__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 \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.182.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

#### 5.182.2 Member Function Documentation

##### 5.182.2.1 `_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()`.



**5.182.2.2** 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.

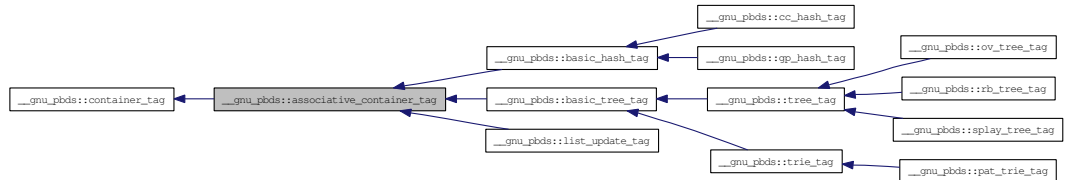
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.183 `__gnu_pbds::associative_container_tag` Struct Reference

Basic associative-container. Inheritance diagram for `__gnu_pbds::associative_container_tag`:



### 5.183.1 Detailed Description

Basic associative-container.

Definition at line 102 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::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator > Class Template

<sup>Reference</sup>  
~~5.184~~ `__gnu_pbds::basic_hash_table`< <sup>1307</sup>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**

### 5.184.1 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.

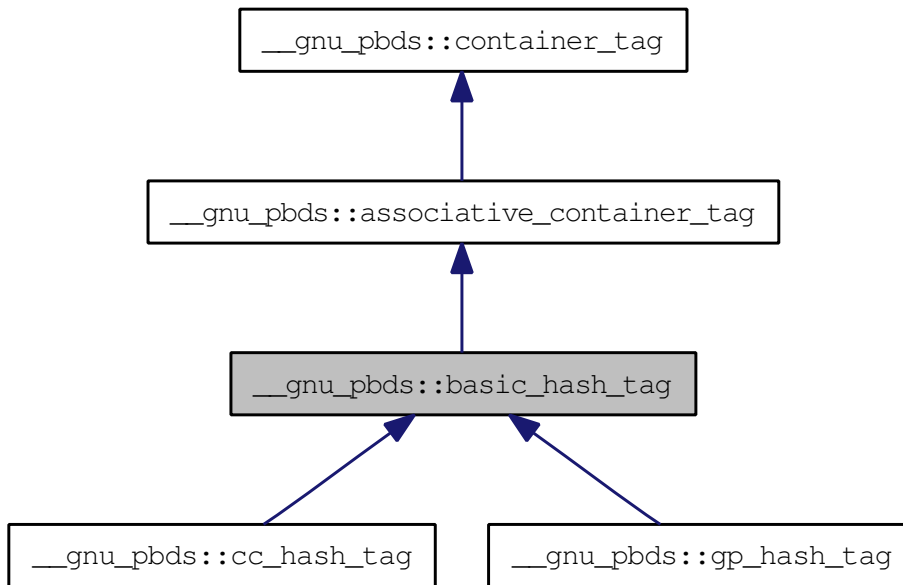
Definition at line 143 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.185 `__gnu_pbds::basic_hash_tag` Struct Reference

Basic hash. Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



### 5.185.1 Detailed Description

Basic hash.

Definition at line 105 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::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**

### **5.186.1 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.

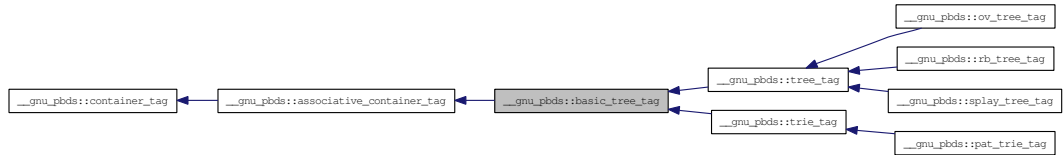
Definition at line 476 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.187 `__gnu_pbds::basic_tree_tag` Struct Reference

Basic [tree](#). Inheritance diagram for `__gnu_pbds::basic_tree_tag`:



### 5.187.1 Detailed Description

Basic [tree](#).

Definition at line 114 of file `tag_and_trait.hpp`.

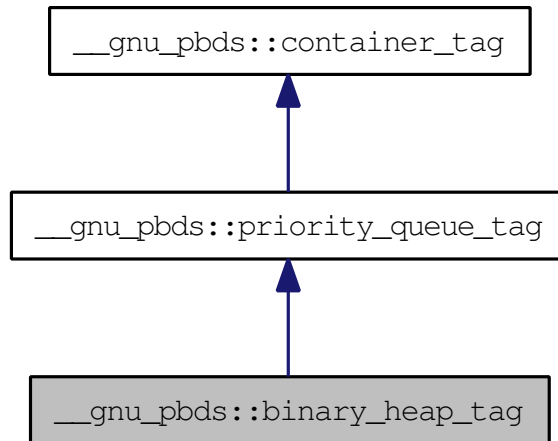
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



## 5.188 `__gnu_pbds::binary_heap_tag` Struct Reference

Binary-heap (array-based). Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



### 5.188.1 Detailed Description

Binary-heap (array-based).

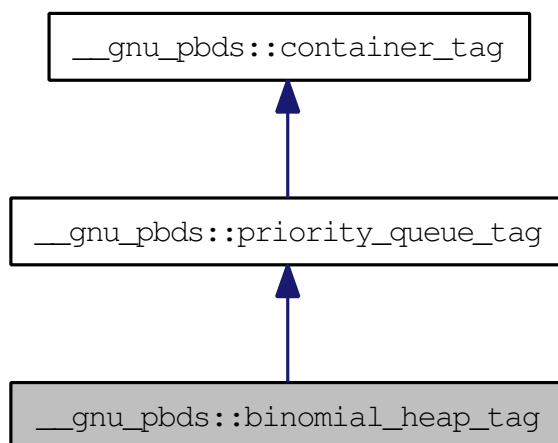
Definition at line 150 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.189 `__gnu_pbds::binomial_heap_tag` Struct Reference

Binomial-heap. Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



### 5.189.1 Detailed Description

Binomial-heap.

Definition at line 144 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

**5.190** `__gnu_pbds::cc_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`, `Comb_Hash_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` > Class Template

**5.190** `__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 [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)
- template<typename It >  
**cc\_hash\_table** (It first, It last, const hash\_fn &h, const eq\_fn &e)
- 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, const comb\_hash\_fn &ch, const resize\_policy &rp)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e)
- **cc\_hash\_table** (const hash\_fn &h)
- [cc\\_hash\\_table](#) & **operator=** (const [cc\\_hash\\_table](#) &other)
- void **swap** ([cc\\_hash\\_table](#) &other)

### 5.190.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name 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, Al-
locator >
```

A concrete collision-chaining hash-based associative container.

Definition at line 179 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

**5.190 `__gnu_pbds::cc_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`,  
`Comb_Hash_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` > Class Template**  
**Reference**

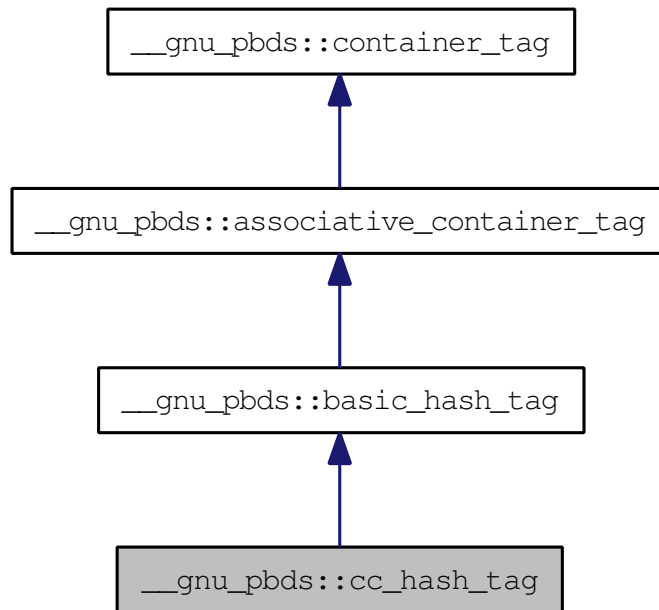
---

**1317**

- [assoc\\_container.hpp](#)

## 5.191 `__gnu_pbds::cc_hash_tag` Struct Reference

Collision-chaining hash. Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



### 5.191.1 Detailed Description

Collision-chaining hash.

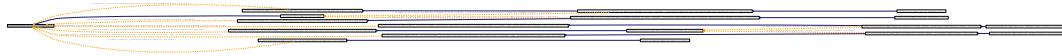
Definition at line 108 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.192 `__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**

### 5.192.1 Detailed Description

```
template<typename Key, typename Mapped, typename Tag, typename Policy_ -
TI, typename Allocator> class __gnu_pbds::container_base< Key, Mapped, Tag,
Policy_TI, Allocator >
```

An abstract basic associative container.

Definition at line 76 of file `assoc_container.hpp`.

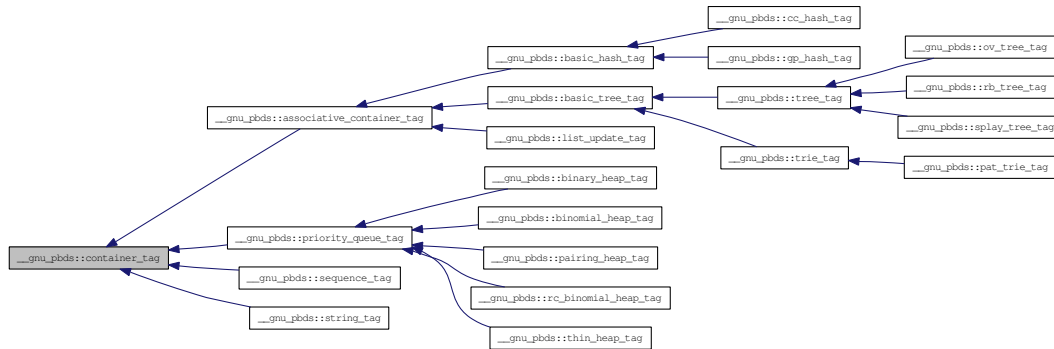
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)



## 5.193 `__gnu_pbds::container_tag` Struct Reference

Base data structure tag. Inheritance diagram for `__gnu_pbds::container_tag`:



### 5.193.1 Detailed Description

Base data structure tag.

Definition at line 92 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.194 `__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**

### 5.194.1 Detailed Description

```
template<typename Cntnr> struct __gnu_pbds::container_traits< Cntnr >
```

[container\\_traits](#)

Definition at line 345 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::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`

### 5.195.1 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](#)

## 5.196 `__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`

### 5.196.1 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](#)

5.197 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >` Struct Template Reference 1325

---

## 5.197 `__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**

#### 5.197.1 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](#)

## 5.198 `__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**

#### 5.198.1 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](#)

5.199 `__gnu_pbds::gp_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`,  
`Comb_Probe_Fn`, `Probe_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` > `Class`  
 Template Reference  
 5.199 `__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. 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 [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)
- template<typename It >  
**gp\_hash\_table** (It first, It last, const hash\_fn &h)
- 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, const comb\_probe\_fn &cp, const probe\_fn &p)
- **gp\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- **gp\_hash\_table** (const hash\_fn &h, const eq\_fn &e)
- **gp\_hash\_table** (const hash\_fn &h)
- [gp\\_hash\\_table](#) & **operator=** (const [gp\\_hash\\_table](#) &other)
- void **swap** ([gp\\_hash\\_table](#) &other)



**5.199 \_\_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  
**5.199.1 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.

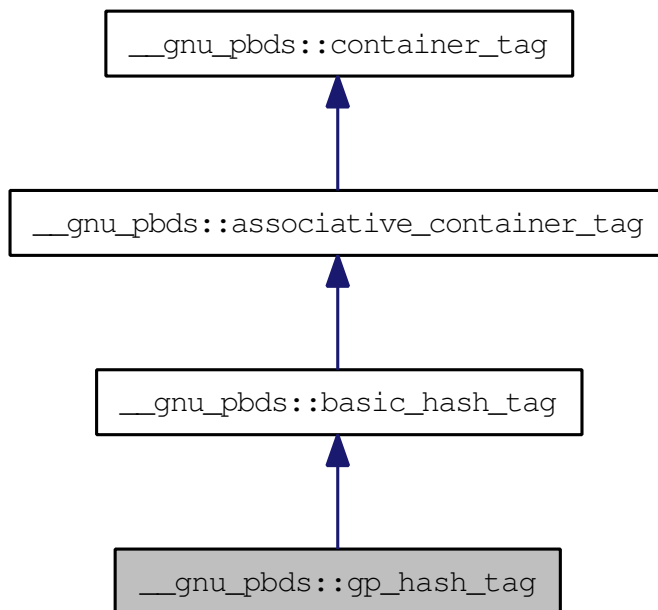
Definition at line 317 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.200 `__gnu_pbds::gp_hash_tag` Struct Reference

General-probing hash. Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



### 5.200.1 Detailed Description

General-probing hash.

Definition at line 111 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::list_update`< Key, Mapped, Eq\_Fn, Update\_Policy, Allocator > **Class Template Reference** 1331

---

## **5.201** `__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

- **list\_update** (const [list\\_update](#) &other)
- template<typename It >  
**list\_update** (It first, It last)
- [list\\_update](#) & **operator=** (const [list\\_update](#) &other)
- void **swap** ([list\\_update](#) &other)

### 5.201.1 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.

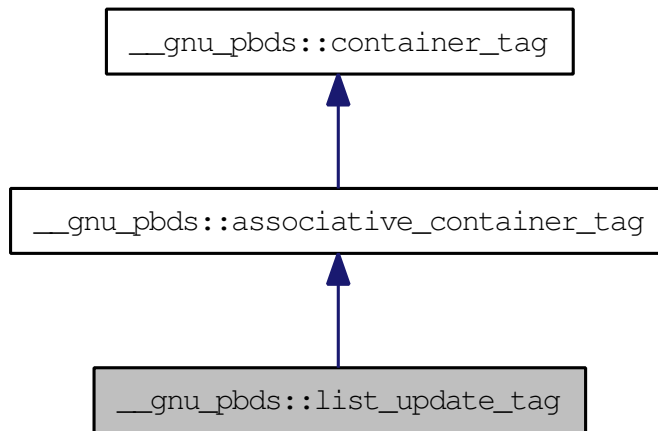
Definition at line 653 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.202 \_\_gnu\_pbds::list\_update\_tag Struct Reference

List-update. Inheritance diagram for \_\_gnu\_pbds::list\_update\_tag:



### 5.202.1 Detailed Description

List-update.

Definition at line 135 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::null_mapped_type` Struct Reference

A mapped-policy indicating that an associative container is a set.

### 5.203.1 Detailed Description

A mapped-policy indicating that an associative container is a set.

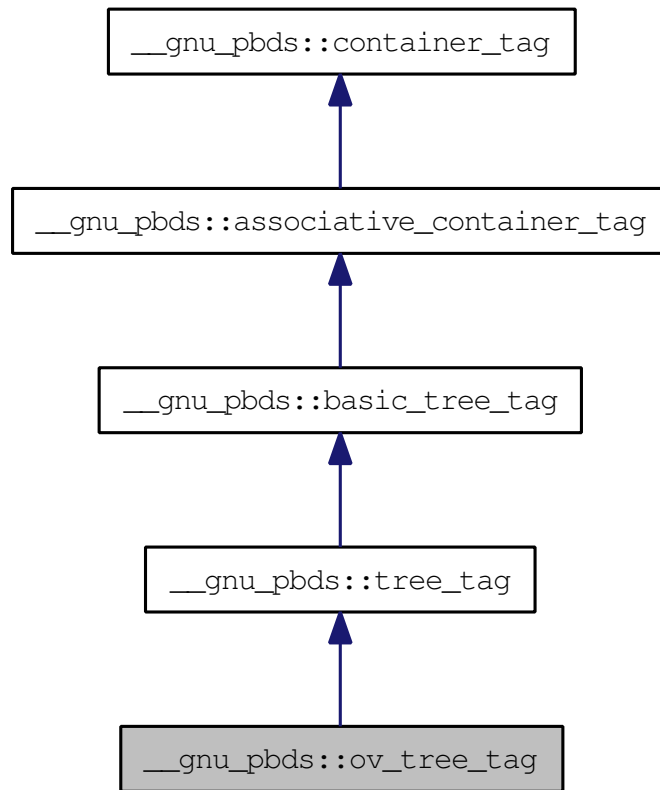
Definition at line 88 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.204 `__gnu_pbds::ov_tree_tag` Struct Reference

Ordered-vector [tree](#). Inheritance diagram for `__gnu_pbds::ov_tree_tag`:



### 5.204.1 Detailed Description

Ordered-vector [tree](#).

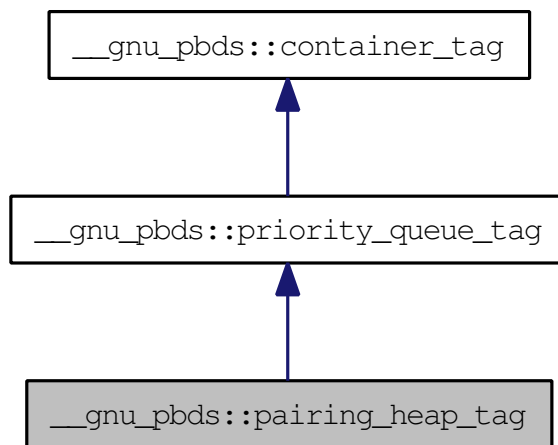
Definition at line 126 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.205 `__gnu_pbds::pairing_heap_tag` Struct Reference

Pairing-heap. Inheritance diagram for `__gnu_pbds::pairing_heap_tag`:



### 5.205.1 Detailed Description

Pairing-heap.

Definition at line 141 of file `tag_and_trait.hpp`.

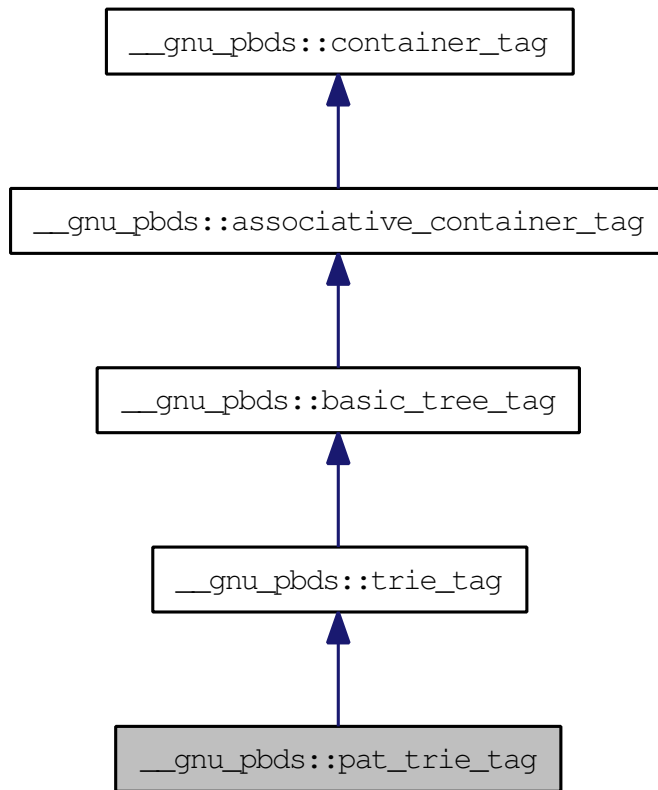
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



## 5.206 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

PATRICIA [trie](#). Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



### 5.206.1 Detailed Description

PATRICIA [trie](#).

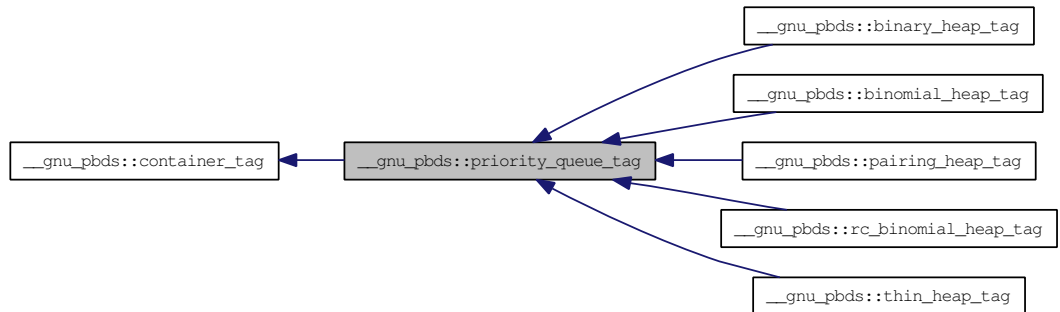
Definition at line 132 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.207 `__gnu_pbds::priority_queue_tag` Struct Reference

Basic priority-queue. Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



### 5.207.1 Detailed Description

Basic priority-queue.

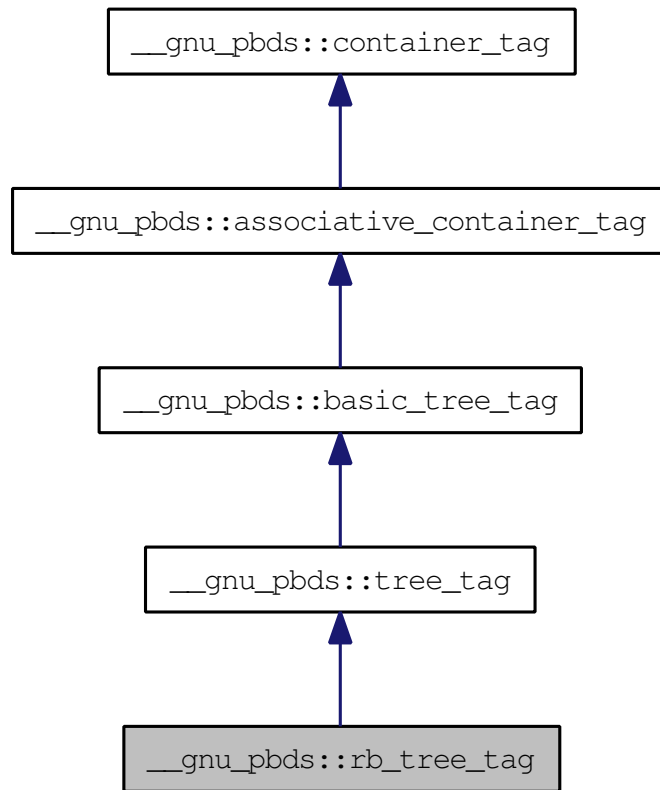
Definition at line 138 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.208 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Red-black [tree](#). Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



### 5.208.1 Detailed Description

Red-black [tree](#).

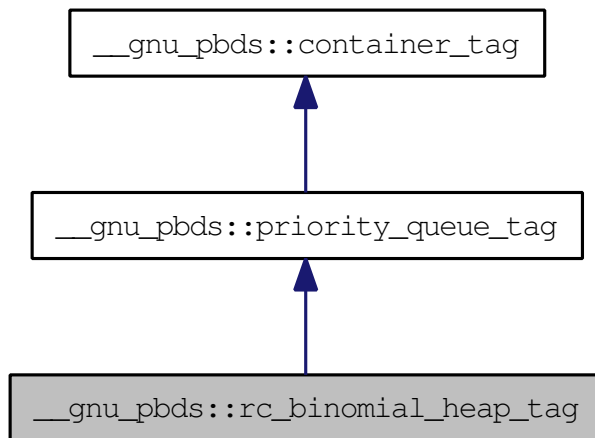
Definition at line 120 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.209 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Redundant-counter binomial-heap. Inheritance diagram for `__gnu_pbds::rc_binomial_heap_tag`:



### 5.209.1 Detailed Description

Redundant-counter binomial-heap.

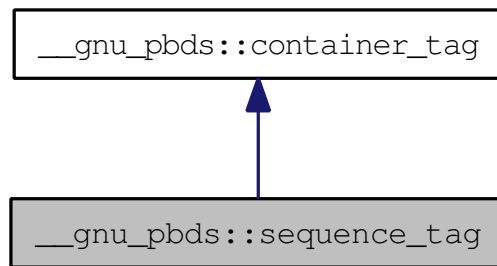
Definition at line 147 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.210 \_\_gnu\_pbds::sequence\_tag Struct Reference

Basic sequence. Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



### 5.210.1 Detailed Description

Basic sequence.

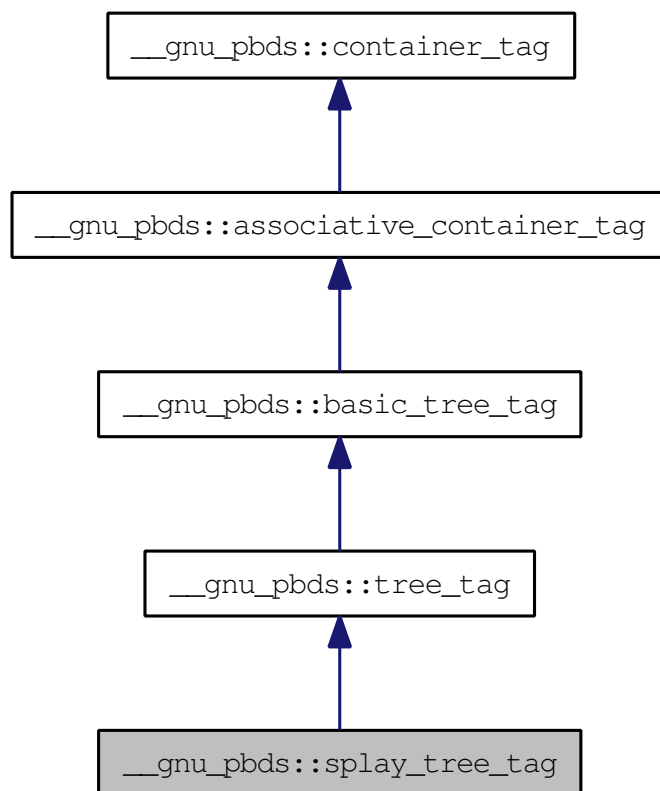
Definition at line 99 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.211 `__gnu_pbds::splay_tree_tag` Struct Reference

Splay [tree](#). Inheritance diagram for `__gnu_pbds::splay_tree_tag`:



### 5.211.1 Detailed Description

Splay [tree](#).

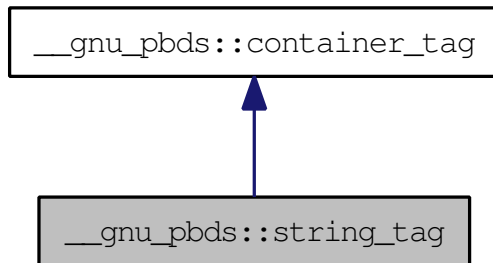
Definition at line 123 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.212 \_\_gnu\_pbds::string\_tag Struct Reference

Basic string container, inclusive of strings, ropes, etc. Inheritance diagram for \_\_gnu\_pbds::string\_tag:



### 5.212.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

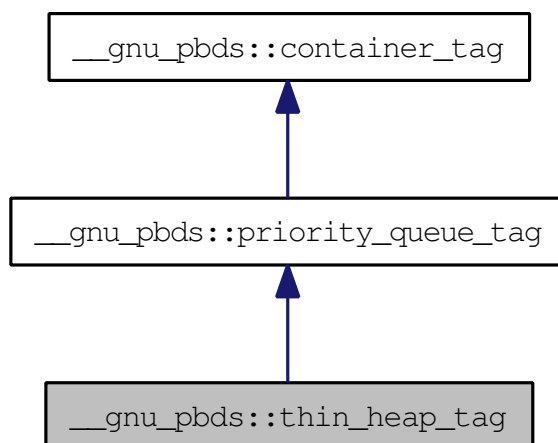
Definition at line 96 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.213 `__gnu_pbds::thin_heap_tag` Struct Reference

Thin heap. Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



### 5.213.1 Detailed Description

Thin heap.

Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



## 5.214 `__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 [tree](#) &other)
- template<typename It >  
**tree** (It first, It last, const cmp\_fn &c)
- template<typename It >  
**tree** (It first, It last)
- **tree** (const cmp\_fn &c)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

### 5.214.1 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.

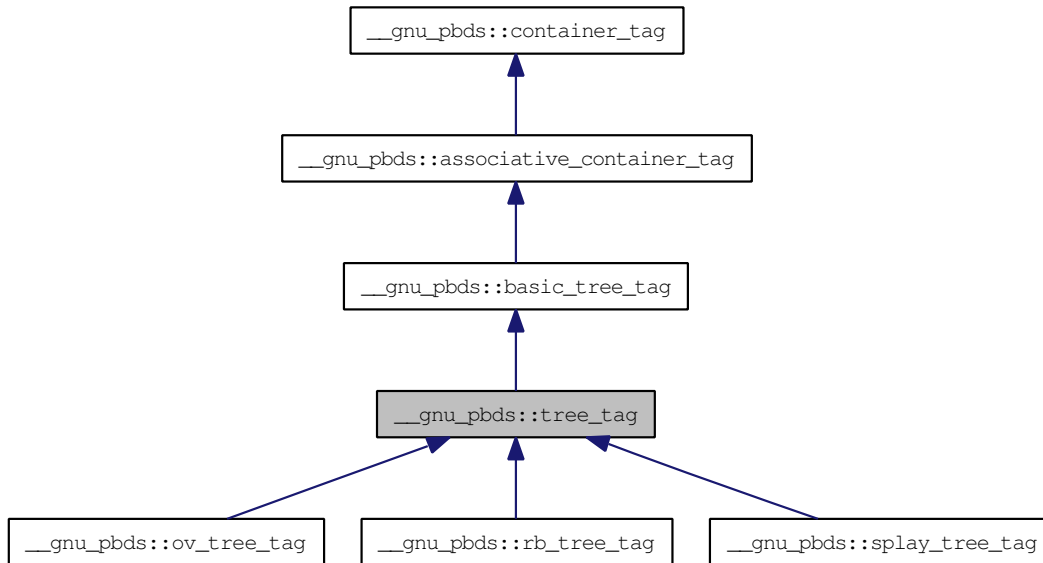
Definition at line 509 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.215 `__gnu_pbds::tree_tag` Struct Reference

[tree](#). Inheritance diagram for `__gnu_pbds::tree_tag`:



### 5.215.1 Detailed Description

[tree](#).

Definition at line 117 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.216 `__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 [trie](#) &other)
- `template`<typename It >  
`trie` (It first, It last, const `e_access_traits` &t)
- `template`<typename It >  
`trie` (It first, It last)
- `trie` (const `e_access_traits` &t)
- `trie` & `operator=` (const [trie](#) &other)
- void `swap` ([trie](#) &other)

### 5.216.1 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.

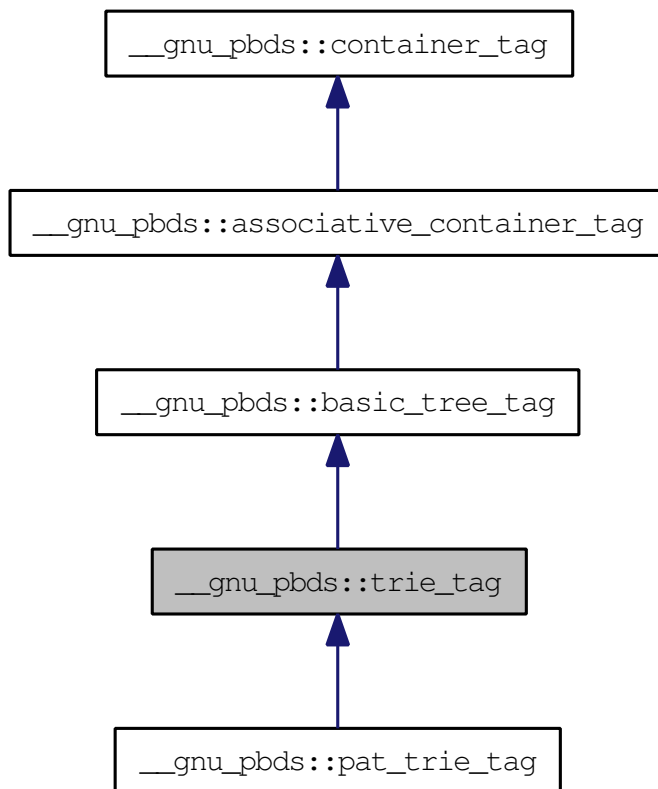
Definition at line 585 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.217 `__gnu_pbds::trie_tag` Struct Reference

[trie](#). Inheritance diagram for `__gnu_pbds::trie_tag`:



### 5.217.1 Detailed Description

[trie](#).

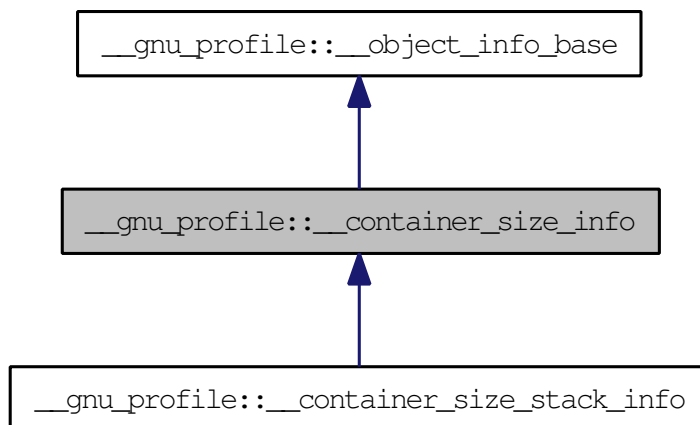
Definition at line 129 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.218 `__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` (`__stack_t __stack`, `size_t __num`)
- `__container_size_info` (`const __container_size_info &__o`)
- `const char * __advice` () const
- `void __destruct` (`size_t __num`, `size_t __inum`)
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (`const __container_size_info &__o`)
- `void __resize` (`size_t __from`, `size_t __to`)
- `float __resize_cost` (`size_t __from`, `size_t __to`)
- `__stack_t __stack` () const
- `void __write` (`FILE *f`) const

### Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

### 5.218.1 Detailed Description

A container size instrumentation line in the object table.

Definition at line 60 of file profiler\_container\_size.h.

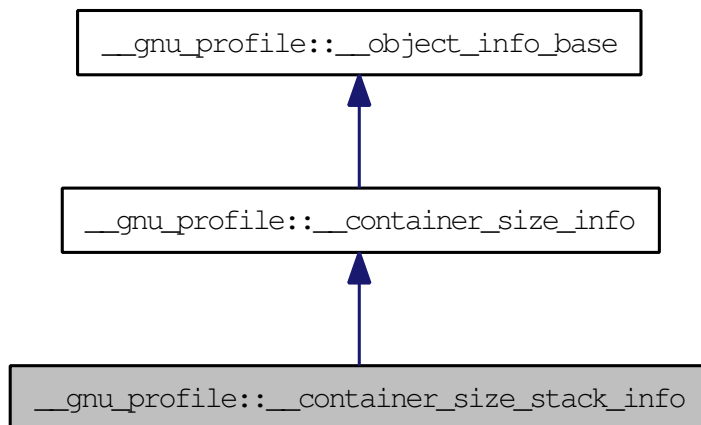
The documentation for this class was generated from the following file:

- profiler\_container\_size.h



## 5.219 `__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`)
- `const char * __advice () const`
- `void __destruct (size_t __num, size_t __inum)`
- `bool __is_valid () const`
- `float __magnitude () const`
- `void __merge (const __container_size_info &_o)`
- `void __resize (size_t __from, size_t __to)`
- `float __resize_cost (size_t __from, size_t __to)`
- `__stack_t __stack () const`
- `void __write (FILE *f) const`

### Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

### 5.219.1 Detailed Description

A container size instrumentation line in the stack table.

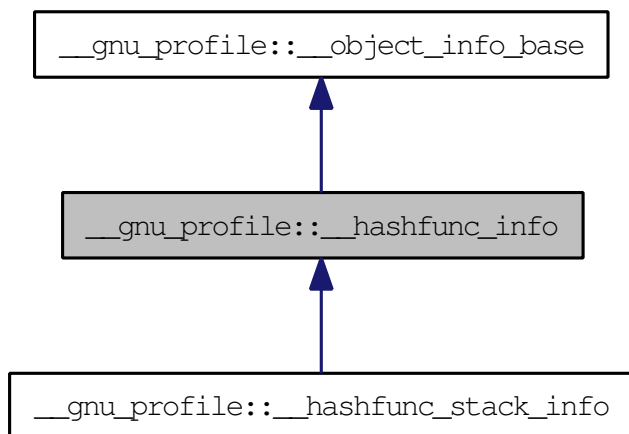
Definition at line 176 of file profiler\_container\_size.h.

The documentation for this class was generated from the following file:

- profiler\_container\_size.h

## 5.220 `__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` (`__stack_t __stack`)
- `__hashfunc_info` (`const __hashfunc_info &o`)
- `const char * __advice` () const
- `void __destruct` (`size_t __chain`, `size_t __accesses`, `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`

#### 5.220.1 Detailed Description

A hash performance instrumentation line in the object table.

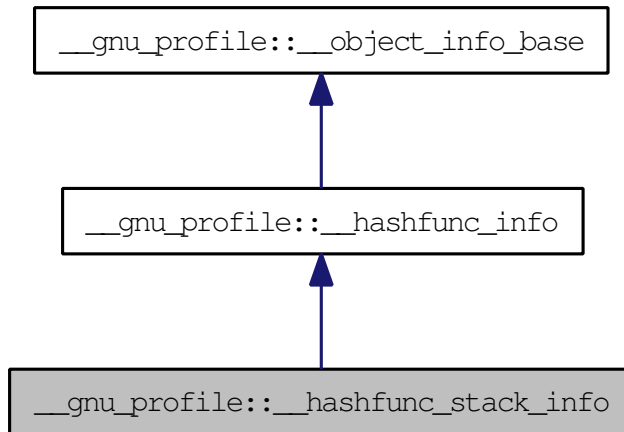
Definition at line 57 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- profiler\_hash\_func.h

## 5.221 `__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`)
- `const char * __advice` () const
- `void __destruct` (size\_t `__chain`, size\_t `__accesses`, 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`

#### 5.221.1 Detailed Description

A hash performance instrumentation line in the stack table.

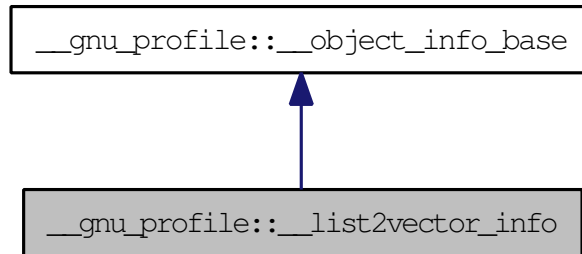
Definition at line 104 of file profiler\_hash\_func.h.

The documentation for this class was generated from the following file:

- profiler\_hash\_func.h

## 5.222 `__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` (const `__list2vector_info` &\_\_o)
- `__list2vector_info` (`__stack_t` \_\_stack)
- const char \* `__advice` () const
- bool `__is_valid` () const
- bool `__is_valid` ()
- size\_t `__iterate` ()
- float `__list_cost` ()
- float `__magnitude` () const
- void `__merge` (const `__list2vector_info` &\_\_o)
- void `__opr_insert` (size\_t \_\_shift, size\_t \_\_size)
- void `__opr_iterate` (size\_t \_\_num)
- void `__resize` (size\_t \_\_from, size\_t \_\_to)
- size\_t `__resize` ()
- void `__set_invalid` ()
- void `__set_list_cost` (float \_\_lc)
- void `__set_vector_cost` (float \_\_vc)
- size\_t `__shift_count` ()
- `__stack_t` `__stack` () const
- void `__write` (FILE \* \_\_f) const

### Protected Attributes

- `__stack_t` `_M_stack`

### 5.222.1 Detailed Description

A list-to-vector instrumentation line in the object table.

Definition at line 59 of file profiler\_list\_to\_vector.h.

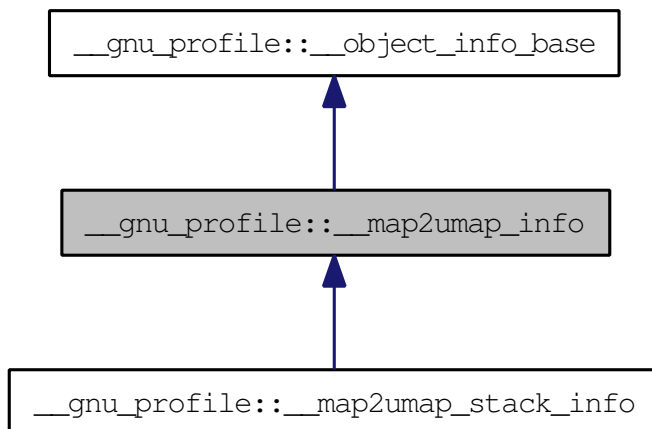
The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)



## 5.223 `__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` (const `__map2umap_info` &o)
- `__map2umap_info` (`__stack_t` \_\_stack)
- `const char * __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__map2umap_info` &o)
- `void __record_erase` (size\_t \_\_size, size\_t \_\_count)
- `void __record_find` (size\_t \_\_size)
- `void __record_insert` (size\_t \_\_size, size\_t \_\_count)
- `void __record_invalidate` ()
- `void __record_iterate` (size\_t \_\_count)
- `__stack_t __stack` () const
- `void __write` (FILE \* \_\_f) const

### Protected Attributes

- `__stack_t _M_stack`

### 5.223.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

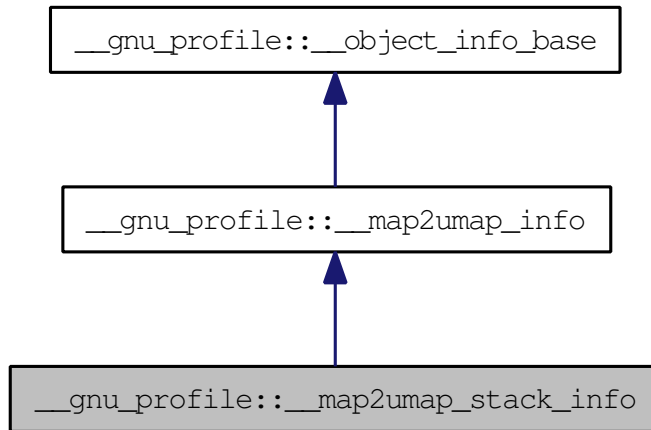
Definition at line 86 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.224 `__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)
- `const char * __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__map2umap_info` &o)
- `void __record_erase` (size\_t \_\_size, size\_t \_\_count)
- `void __record_find` (size\_t \_\_size)
- `void __record_insert` (size\_t \_\_size, size\_t \_\_count)
- `void __record_invalidate` ()
- `void __record_iterate` (size\_t \_\_count)
- `__stack_t __stack` () const
- `void __write` (FILE \*\_\_f) const

### Protected Attributes

- `__stack_t _M_stack`

### 5.224.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

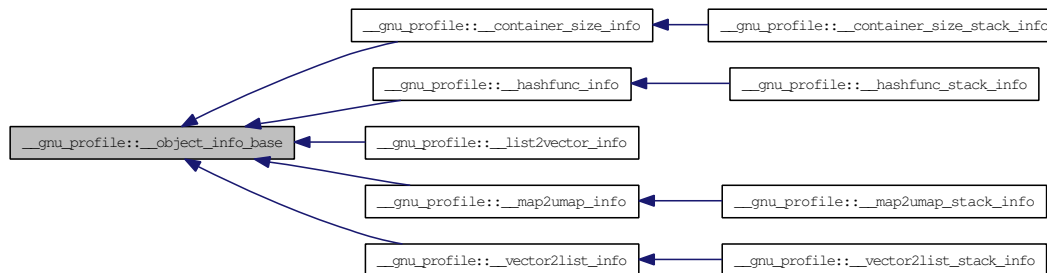
Definition at line 189 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.225 `__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` (const `__object_info_base` &o)
- `__object_info_base` (`__stack_t` \_\_stack)
- `bool __is_valid` () const
- `__stack_t __stack` () const
- virtual void `__write` (FILE \*f) const =0

### Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

#### 5.225.1 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](#)

## 5.226 `__gnu_profile::__reentrance_guard` Struct Reference

Reentrance guard.

### Static Public Member Functions

- static bool `__get_in` ()
- static bool & `__inside` ()

#### 5.226.1 Detailed Description

Reentrance guard. Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 62 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

## 5.227 `__gnu_profile::__stack_hash` Class Reference

Hash function for summary trace using call stack as index.

### Public Member Functions

- `bool operator() (const __stack_t __stack1, const __stack_t __stack2) const`
- `size_t operator() (const __stack_t __s) const`

#### 5.227.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 101 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.228 `__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

### 5.228.1 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 160 of file `profiler_node.h`.

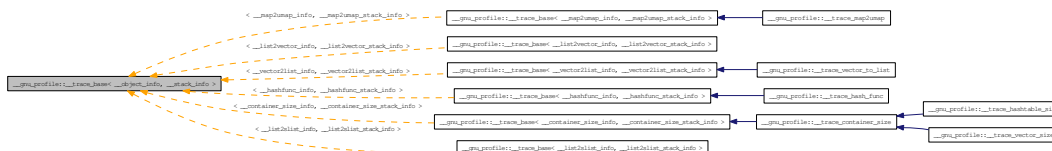
The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)



## 5.229 `__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 `__lock_object_table` ()
- void `__lock_stack_table` ()
- void `__retire_object` (`__object_t` \_\_object)
- void `__unlock_object_table` ()
- void `__unlock_stack_table` ()
- void `__write` (FILE \*f)

### Protected Attributes

- const char \* `__id`

#### 5.229.1 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 228 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

## 5.230 `__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, size\_t \_\_inum)
- void **\_\_destruct** (const void \*\_\_obj, size\_t \_\_num, size\_t \_\_inum)
- `__container_size_info *` **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (const \_\_object\_t \_\_obj, \_\_stack\_t \_\_stack, size\_t \_\_num)
- void **\_\_lock\_object\_table** ()
- void **\_\_lock\_stack\_table** ()
- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_unlock\_object\_table** ()
- void **\_\_unlock\_stack\_table** ()
- void **\_\_write** (FILE \*f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.230.1 Detailed Description

Container size instrumentation trace producer.

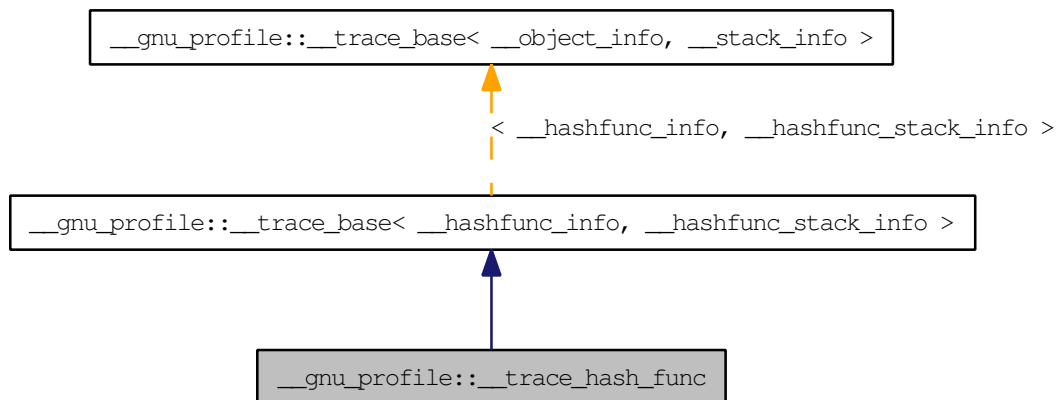
Definition at line 184 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- `profiler_container_size.h`

## 5.231 \_\_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, size\_t \_\_chain, size\_t \_\_accesses, size\_t \_\_hops)
- **\_\_hashfunc\_info** \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_lock\_object\_table** ()
- void **\_\_lock\_stack\_table** ()
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_unlock\_object\_table** ()
- void **\_\_unlock\_stack\_table** ()
- void **\_\_write** (FILE \*f)

### Protected Attributes

- const char \* **\_\_id**

### 5.231.1 Detailed Description

Hash performance instrumentation producer.

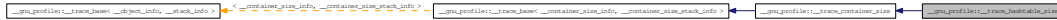
Definition at line 110 of file profiler\_hash\_func.h.

The documentation for this class was generated from the following file:

- profiler\_hash\_func.h

## 5.232 `__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`, `size_t` `__inum`)
- void `__destruct` (const void \*`__obj`, `size_t` `__num`, `size_t` `__inum`)
- `__container_size_info` \* `__get_object_info` (`__object_t` `__object`)
- void `__insert` (const `__object_t` `__obj`, `__stack_t` `__stack`, `size_t` `__num`)
- void `__lock_object_table` ()
- void `__lock_stack_table` ()
- void `__resize` (const void \*`__obj`, int `__from`, int `__to`)
- void `__retire_object` (`__object_t` `__object`)
- void `__unlock_object_table` ()
- void `__unlock_stack_table` ()
- void `__write` (FILE \*`f`)

### Protected Attributes

- const char \* `__id`

#### 5.232.1 Detailed Description

Hashtable size instrumentation trace producer.

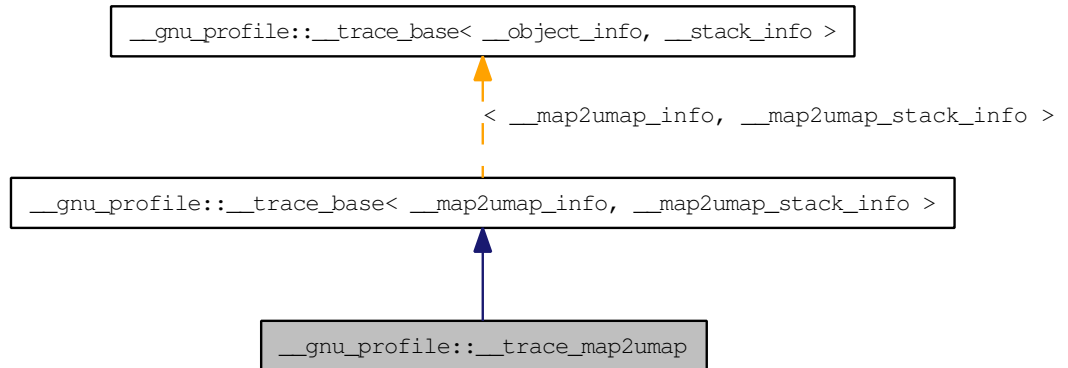
Definition at line 59 of file `profiler_hashtable_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

## 5.233 `__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 `__lock_object_table` ()
- void `__lock_stack_table` ()
- void `__retire_object` (`__object_t` \_\_object)
- void `__unlock_object_table` ()
- void `__unlock_stack_table` ()
- void `__write` (FILE \*f)

### Protected Attributes

- const char \* `__id`

#### 5.233.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

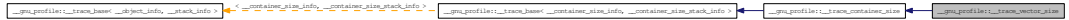
Definition at line 196 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.234 `__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, size\_t \_\_inum)
- void **\_\_destruct** (const void \*\_\_obj, size\_t \_\_num, size\_t \_\_inum)
- `__container_size_info` \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (const \_\_object\_t \_\_obj, \_\_stack\_t \_\_stack, size\_t \_\_num)
- void **\_\_lock\_object\_table** ()
- void **\_\_lock\_stack\_table** ()
- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_unlock\_object\_table** ()
- void **\_\_unlock\_stack\_table** ()
- void **\_\_write** (FILE \*f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.234.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 59 of file `profiler_vector_size.h`.

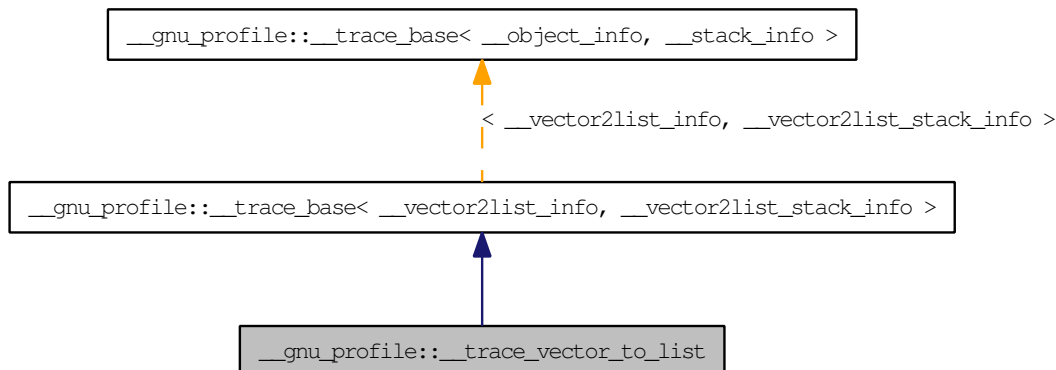
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)



## 5.235 `__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` (`size_t` `__shift`, `size_t` `__iterate`, `size_t` `__resize`)
- void `__lock_object_table` ()
- void `__lock_stack_table` ()
- void `__opr_find` (`const void *``__obj`, `size_t` `__size`)
- void `__opr_insert` (`const void *``__obj`, `size_t` `__pos`, `size_t` `__num`)
- void `__opr_iterate` (`const void *``__obj`, `size_t` `__num`)
- void `__resize` (`const void *``__obj`, `size_t` `__from`, `size_t` `__to`)
- void `__retire_object` (`__object_t` `__object`)
- void `__unlock_object_table` ()
- void `__unlock_stack_table` ()
- float `__vector_cost` (`size_t` `__shift`, `size_t` `__iterate`, `size_t` `__resize`)
- void `__write` (`FILE *``f`)

## Protected Attributes

- const char \* `__id`

### 5.235.1 Detailed Description

Vector-to-list instrumentation producer.

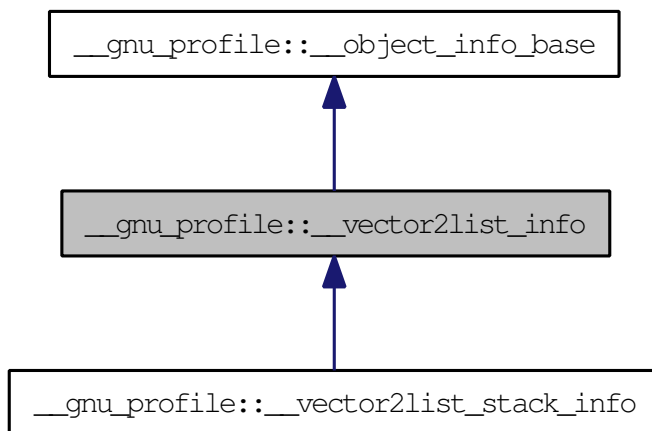
Definition at line 147 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.236 `__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` (const `__vector2list_info` &\_\_o)
- `__vector2list_info` (`__stack_t` \_\_stack)
- `const char * __advice` () const
- `bool __is_valid` () const
- `bool __is_valid` ()
- `size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const `__vector2list_info` &\_\_o)
- `void __opr_find` (size\_t \_\_size)
- `void __opr_insert` (size\_t \_\_pos, size\_t \_\_num)
- `void __opr_iterate` (size\_t \_\_num)
- `void __resize` (size\_t \_\_from, size\_t \_\_to)
- `size_t __resize` ()
- `void __set_invalid` ()
- `void __set_list_cost` (float \_\_lc)
- `void __set_vector_cost` (float \_\_vc)
- `size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (FILE \* \_\_f) const

## Protected Attributes

- `__stack_t_M_stack`

### 5.236.1 Detailed Description

A vector-to-list instrumentation line in the object table.

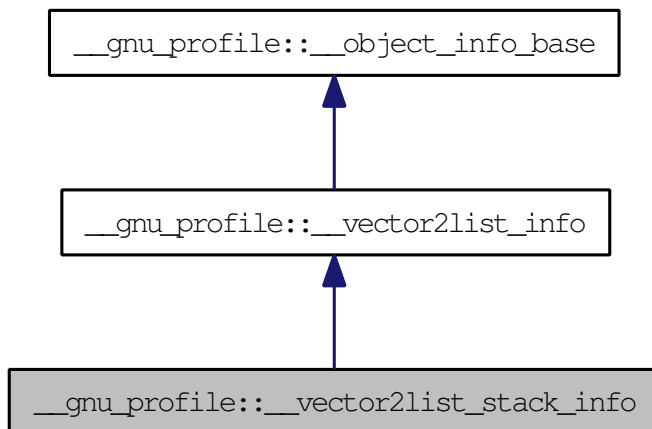
Definition at line 57 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.237 `__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`)
- `const char * __advice` () const
- `bool __is_valid` () const
- `bool __is_valid` ()
- `size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const `__vector2list_info` &`_o`)
- `void __opr_find` (size\_t `_size`)
- `void __opr_insert` (size\_t `_pos`, size\_t `_num`)
- `void __opr_iterate` (size\_t `_num`)
- `void __resize` (size\_t `_from`, size\_t `_to`)
- `size_t __resize` ()
- `void __set_invalid` ()
- `void __set_list_cost` (float `_lc`)
- `void __set_vector_cost` (float `_vc`)
- `size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (FILE \*`_f`) const

### Protected Attributes

- `__stack_t_M_stack`

#### 5.237.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 140 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.238 `__gnu_profile::__warning_data` Struct Reference

Representation of a warning.

### Public Member Functions

- `__warning_data` (float `__m`, `__stack_t` `__c`, const char \*`__id`, const char \*`__msg`)
- bool `operator>` (const struct `__warning_data` &other) const

### Public Attributes

- `__stack_t` `__context`
- float `__magnitude`
- const char \* `__warning_id`
- const char \* `__warning_message`

#### 5.238.1 Detailed Description

Representation of a warning.

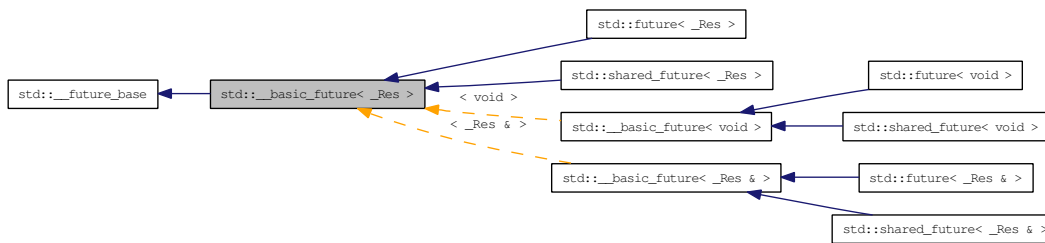
Definition at line 95 of file `profiler_trace.h`.

The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

## 5.239 `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

### Protected Types

- typedef `__future_base::Result`< `_Res` > & `__result_type`
- typedef `shared_ptr`< `_State` > `__state_type`

### Protected Member Functions

- `__basic_future` (`future`< `_Res` > &&)
- `__basic_future` (`shared_future`< `_Res` > &&)
- `__basic_future` (const `shared_future`< `_Res` > &)
- `__basic_future` (const `__state_type` &`__state`)
- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` &`__that`)



### 5.239.1 Detailed Description

`template<typename _Res> class std::__basic_future<_Res>`

Common implementation for [future](#) and [shared\\_future](#).

Definition at line 480 of file `future`.

### 5.239.2 Member Function Documentation

**5.239.2.1** `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 513 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](#)

## 5.240 `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 >`:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- 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

### Protected Member Functions

- `__codecvt_abstract_base` (size\_t \_\_refs=0)
- `__attribute` ((\_\_const\_\_)) static const char \* `_S_get_c_name` () throw ()
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()

## 5.240 `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>` Class Template Reference 1387

---

- 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 \*\_\_from\_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 `__c_locale` **\_S\_lc\_ctype\_c\_locale** (`__c_locale` \_\_cloc, const char \*\_\_s)

### Friends

- class `locale::Impl`

### 5.240.1 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 67 of file `codecvt.h`.

## 5.240.2 Member Function Documentation

**5.240.2.1** `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 >`.

**5.240.2.2** `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 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.240.2.3** `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 115 of file codecvt.h.

**5.240.2.4** `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.

**5.240 std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT > Class**  
**Template Reference** **1391**

---

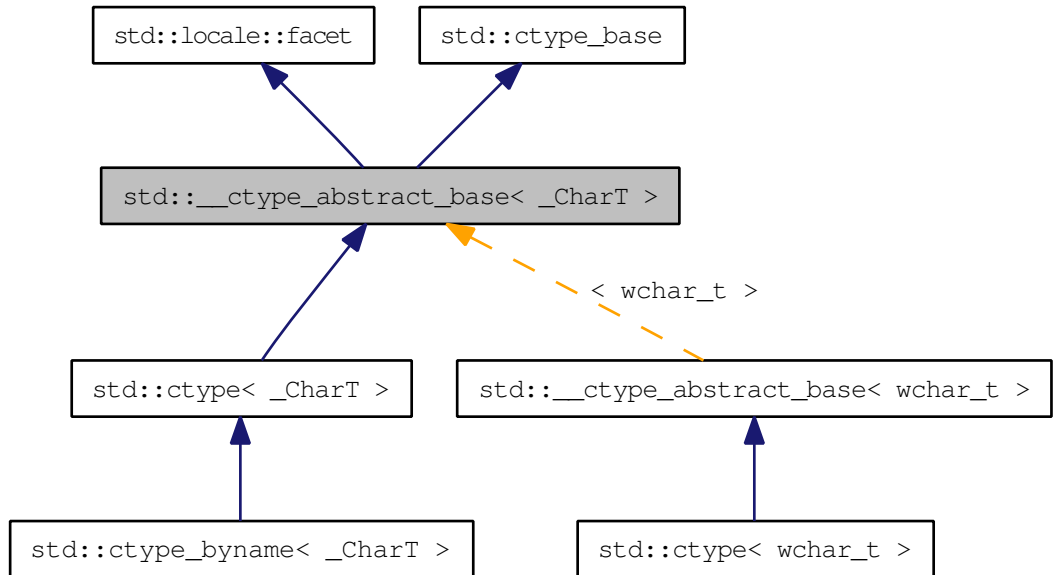
Definition at line 154 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.241 `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

- `const char_type * is (const char_type * __lo, const char_type * __hi, mask * __vec) const`
- `bool is (mask __m, char_type __c) 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
- 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)
- `__attribute__` ((\_\_const\_\_)) static const `char` \* `_S_get_c_name`() throw ()
- 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 const `char_type` \* `do_narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, `char` `__dfault`, `char` \* `__dest`) const =0
- virtual `char` `do_narrow` (`char_type`, `char` `__dfault`) 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 const `char_type` \* `do_tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual `char_type` `do_tolower` (`char_type`) const =0
- virtual const `char_type` \* `do_toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0

- virtual [char\\_type do\\_toupper](#) ([char\\_type](#)) 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 [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) [\\_\\_cloc](#), const char \*\_\_s)

## Friends

- class [locale::\\_Impl](#)

### 5.241.1 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 143 of file [locale\\_facets.h](#).

### 5.241.2 Member Typedef Documentation

**5.241.2.1 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 148 of file [locale\\_facets.h](#).

### 5.241.3 Member Function Documentation

**5.241.3.1** `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>`.

**5.241.3.2** `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:**

$(M \& m) \neq 0$ .

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

**5.241.3.3** `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 >](#).

**5.241.3.4** `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 >](#).

```
5.241.3.5 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>`.

```
5.241.3.6 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>`.

**5.241.3.7** `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 >](#).

**5.241.3.8** `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 >](#).

```
5.241.3.9 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>`.

```
5.241.3.10 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>`.

**5.241.3.11** `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 >](#).

**5.241.3.12** `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 >](#).

---



**5.241.3.13** `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 178 of file locale\_facets.h.

**5.241.3.14** `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:**

$(M \& m) \neq 0$ .

Definition at line 161 of file locale\_facets.h.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.241.3.15** `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 [codecv](#) 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 345 of file `locale_facets.h`.

**5.241.3.16** `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 [codecv](#) for that.

**Parameters:**

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

**Returns:**

The converted `char`.

Definition at line 323 of file `locale_facets.h`.

Referenced by `std::time_put<_CharT, _OutIter >::put()`.

**5.241.3.17** `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 194 of file locale\_facets.h.

```
5.241.3.18 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 210 of file locale\_facets.h.

```
5.241.3.19 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_tolower(lo, hi)`.

**Parameters:**

- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns:**

*hi*.

Definition at line 268 of file locale\_facets.h.

**5.241.3.20** `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_tolower(c)`.

**Parameters:**

*c* The `char_type` to convert.

**Returns:**

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file locale\_facets.h.

**5.241.3.21** `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 239 of file locale\_facets.h.

**5.241.3.22** `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 224 of file `locale_facets.h`.

**5.241.3.23** `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 304 of file `locale_facets.h`.

**5.241.3.24** `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 [codecv](#) for that.

**Parameters:**

*c* The char to convert.

**Returns:**

The converted `char_type`.

Definition at line 285 of file `locale_facets.h`.

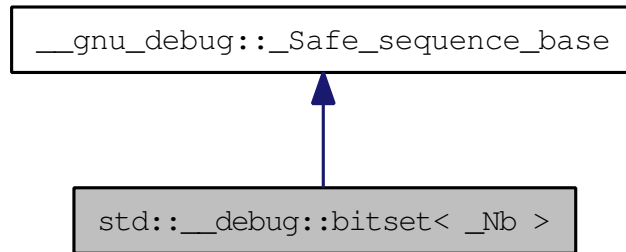
Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::regex_traits< _Ch_type >::isctype()`, and `std::operator<<()`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.242 `std::__debug::bitset< _Nb >` Class Template Reference

Class `std::bitset` with additional safety/checking/debug instrumentation. Inheritance diagram for `std::__debug::bitset< _Nb >`:



### Public Member Functions

- **bitset** (const char \* \_\_str)
- **bitset** (const `_Base` & \_\_x)
- 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')`)
- 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))
- **bitset** (unsigned long long \_\_val)
- const `_Base` & **\_M\_base** () const
- `_Base` & **\_M\_base** ()
- void **\_M\_invalidate\_all** () const
- `bitset`< `_Nb` > & **flip** (size\_t \_\_pos)
- `bitset`< `_Nb` > & **flip** ()
- 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)
- bool [operator](#)[] (size\_t \_\_pos) const
- reference [operator](#)[] (size\_t \_\_pos)
- [bitset](#)< \_Nb > & [operator](#)^= (const [bitset](#)< \_Nb > &\_\_rhs)
- [bitset](#)< \_Nb > & [operator](#)|= (const [bitset](#)< \_Nb > &\_\_rhs)
- [bitset](#)< \_Nb > [operator](#)~ () const
- [bitset](#)< \_Nb > & [reset](#) (size\_t \_\_pos)
- [bitset](#)< \_Nb > & [reset](#) ()
- [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
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [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<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [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<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) () 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)

### 5.242.1 Detailed Description

`template<size_t _Nb> class std::__debug::bitset< _Nb >`

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

Definition at line 43 of file `debug/bitset`.

### 5.242.2 Member Function Documentation

**5.242.2.1** void [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_detach\\_all](#) ()  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.242.2.2** 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`.

**5.242.2.3** [\\_\\_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\\_iter](#)()

**5.242.2.4** void [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_invalidate\\_all](#) () const  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

#### 5.242.2.5 `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).

#### 5.242.2.6 `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.

### 5.242.3 Member Data Documentation

#### 5.242.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

#### 5.242.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 5.242.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` `[mutable, inherited]`

The container version number. This number may never be 0.

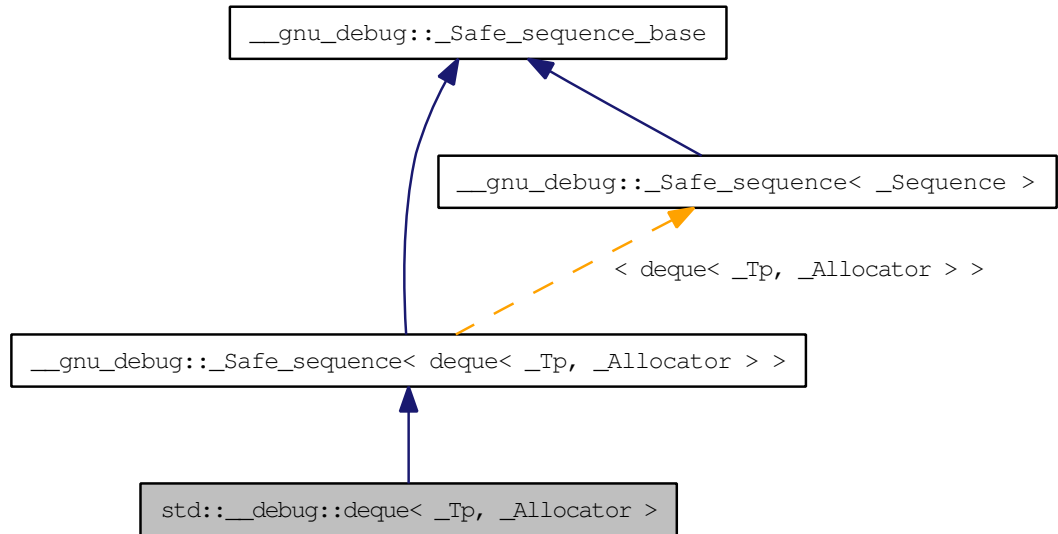
Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 5.243 `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< typename _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< typename _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** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **deque** ([deque](#) &&\_\_x)
- **deque** (const [\\_Base](#) &\_\_x)
- **deque** (const [deque](#) &\_\_x)
- template<class [\\_InputIterator](#) >  
**deque** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** (size\_type \_\_n, const [\\_Tp](#) &\_\_value=[\\_Tp](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** (const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [\\_Base](#) & [\\_M\\_base](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [deque](#)< [\\_Tp](#), [\\_Allocator](#) > > &\_\_x)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- void **assign** (size\_type \_\_n, const [\\_Tp](#) &\_\_t)
- template<class [\\_InputIterator](#) >  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- 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)
- template<typename... [\\_Args](#) >  
void **emplace\_back** ([\\_Args](#) &&...\_\_args)
- template<typename... [\\_Args](#) >  
void **emplace\_front** ([\\_Args](#) &&...\_\_args)
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- [iterator](#) **erase** ([iterator](#) \_\_first, [iterator](#) \_\_last)
- [iterator](#) **erase** ([iterator](#) \_\_position)
- const\_reference **front** () const

- reference **front** ()
- template<class [\\_InputIterator](#) >  
void **insert** ([iterator](#) \_\_position, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **insert** ([iterator](#) \_\_position, [size\\_type](#) \_\_n, const [\\_Tp](#) &\_\_x)
- void **insert** ([iterator](#) \_\_p, [initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [iterator](#) **insert** ([iterator](#) \_\_position, [\\_Tp](#) &&\_\_x)
- [iterator](#) **insert** ([iterator](#) \_\_position, const [\\_Tp](#) &\_\_x)
- [deque](#) & **operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [deque](#) & **operator=** ([deque](#) &&\_\_x)
- [deque](#) & **operator=** (const [deque](#) &\_\_x)
- const\_reference **operator[]** ([size\\_type](#) \_\_n) const
- reference **operator[]** ([size\\_type](#) \_\_n)
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** ([\\_Tp](#) &&\_\_x)
- void **push\_back** (const [\\_Tp](#) &\_\_x)
- void **push\_front** ([\\_Tp](#) &&\_\_x)
- void **push\_front** (const [\\_Tp](#) &\_\_x)
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- [reverse\\_iterator](#) **rend** ()
- void **resize** ([size\\_type](#) \_\_sz, [\\_Tp](#) \_\_c=[\\_Tp](#)())
- void **swap** ([deque](#) &\_\_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)

### 5.243.1 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`.

### 5.243.2 Member Function Documentation

5.243.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

5.243.2.2 `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`.

5.243.2.3 `__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_iter()`.

5.243.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const` [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string<_`

`CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator+=(())`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::push_back()`.

**5.243.2.5** `void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::M_invalidate_if(_Predicate _pred) [inline, inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.243.2.6** `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).

**5.243.2.7** `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.

**5.243.2.8** `void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::M_transfer_iter(const _Safe_iterator< _Iterator, deque< _Tp, _Allocator > > & _x) [inline, inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.243.3 Member Data Documentation

**5.243.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.



## **5.243 std::\_\_debug::deque< \_Tp, \_Allocator > Class Template Reference 1417**

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

### **5.243.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_iterators [inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

### **5.243.3.3 unsigned int \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version [mutable, inherited]**

The container version number. This number may never be 0.

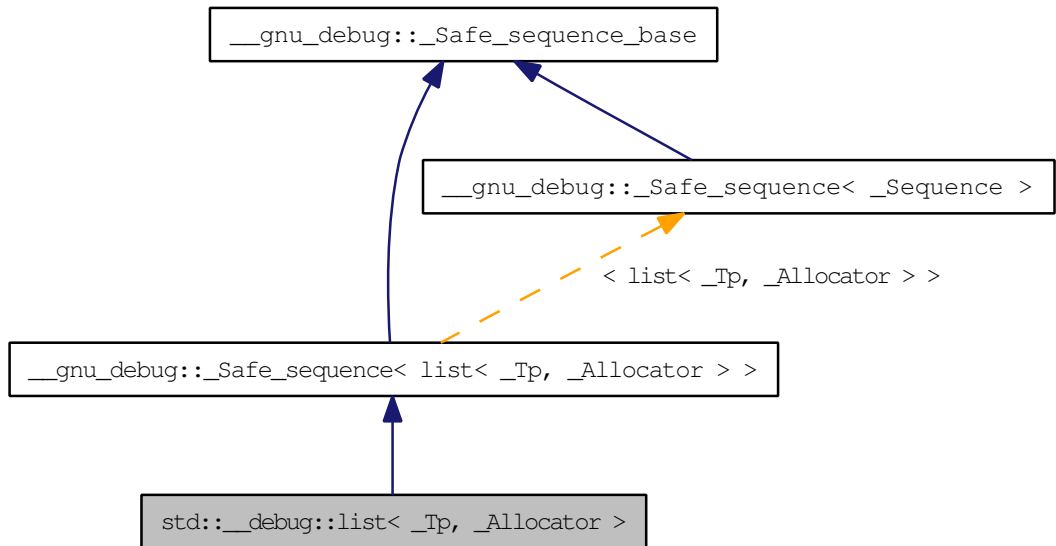
Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/deque](#)

## 5.244 `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< 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 `__gnu_debug::_Safe_iterator< 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** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **list** ([list](#) &&\_\_x)
- **list** (const [\\_Base](#) &\_\_x)
- **list** (const [list](#) &\_\_x)
- template<class [\\_InputIterator](#) >  
**list** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=\_Allocator())
- **list** (size\_type \_\_n, const [\\_Tp](#) &\_\_value=\_Tp(), const [\\_Allocator](#) &\_\_a=\_Allocator())
- **list** (const [\\_Allocator](#) &\_\_a=\_Allocator())
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [\\_Base](#) & [\\_M\\_base](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [list](#)< [\\_Tp](#), [\\_Allocator](#) > &\_\_x)
- void **assign** (size\_type \_\_n, const [\\_Tp](#) &\_\_t)
- template<class [\\_InputIterator](#) >  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- 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)
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- [iterator](#) **erase** ([iterator](#) \_\_position, [iterator](#) \_\_last)
- [iterator](#) **erase** ([iterator](#) \_\_position)
- const\_reference **front** () const
- reference **front** ()
- template<class [\\_InputIterator](#) >  
void **insert** ([iterator](#) \_\_position, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **insert** ([iterator](#) \_\_position, size\_type \_\_n, const [\\_Tp](#) &\_\_x)

- void **insert** (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- **list** & **insert** (iterator \_\_position, \_Tp &&\_\_x)
- **list** & **insert** (iterator \_\_position, const \_Tp &\_\_x)
- template<typename \_Compare >  
void **merge** (list &\_\_x, \_Compare \_\_comp)
- template<class \_Compare >  
void **merge** (list &&\_\_x, \_Compare \_\_comp)
- void **merge** (list &\_\_x)
- void **merge** (list &&\_\_x)
- **list** & **operator=** (initializer\_list< value\_type > \_\_l)
- **list** & **operator=** (list &&\_\_x)
- **list** & **operator=** (const list &\_\_x)
- void **pop\_back** ()
- void **pop\_front** ()
- **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, \_Tp \_\_c=\_Tp())
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **sort** ()
- 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, iterator \_\_i)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x)
- void **swap** (list &\_\_x)
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_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)`

### 5.244.1 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 44 of file `debug/list`.

### 5.244.2 Member Function Documentation

**5.244.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.244.2.2** `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`.

**5.244.2.3** `__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_iter()`.

**5.244.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all () const [inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

Referenced by \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::append(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::assign(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::c\_str(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::clear(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::data(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::operator+=(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::operator=(), and \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::push\_back().

**5.244.2.5 void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator > >::M\_invalidate\_if(\_Predicate \_\_pred) [inline, inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred(x)* returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

**5.244.2.6 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).

**5.244.2.7 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.

**5.244.2.8 void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator > >::M\_transfer\_iter (const \_Safe\_iterator< \_Iterator, list< \_Tp, \_Allocator > > & \_\_x) [inline, inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

#### 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 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

#### 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.

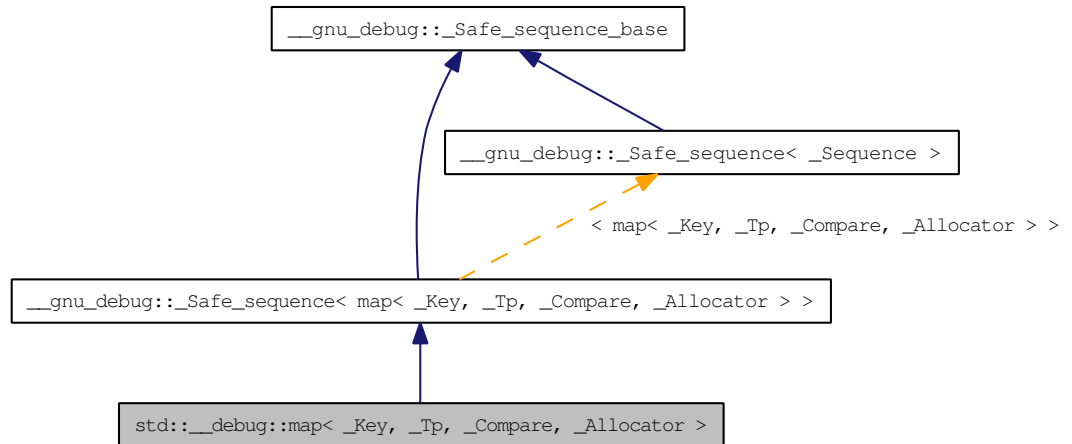
Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/list](#)

## 5.245 `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** (`initializer_list< value_type > __l`, `const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- **map** (`map &&__x`)
- **map** (`const _Base &__x`)
- **map** (`const map &__x`)
- `template<typename _InputIterator >`  
  **map** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **map** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, map< _Key, _Tp, _Compare, _Allocator > > &__x)`
- `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`
- `const_iterator end () const`
- `iterator end ()`
- `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 (iterator __first, iterator __last)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __position)`
- `const_iterator find (const key_type &__x) const`
- `iterator find (const key_type &__x)`
- `template<typename _InputIterator >`  
  `void insert (_InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, const value_type &__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `map & operator= (initializer_list< value_type > __l)`

- `map` & `operator=` (`map` &&\_\_x)
- `map` & `operator=` (`const map` &\_\_x)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- void `swap` (`map` &\_\_x)
- `const_iterator` `upper_bound` (`const key_type` &\_\_x) const
- `iterator` `upper_bound` (`const key_type` &\_\_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)

### 5.245.1 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`.

### 5.245.2 Member Function Documentation

**5.245.2.1** void `__gnu_debug::Safe_sequence_base::_M_detach_all` ()  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.245 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference** 1427

---

**5.245.2.2 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.

**5.245.2.3 \_\_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_iter()`.

**5.245.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all () const**  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::append()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator+=(())`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::push_back()`.

**5.245.2.5 void \_\_gnu\_debug::Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_invalidate\_if (\_Predicate \_\_pred)** [inline, inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.245.2.6** `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).

**5.245.2.7** `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.

**5.245.2.8** `void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare,`  
`_Allocator > >::M_transfer_iter (const Safe_iterator< _Iterator,`  
`map< _Key, _Tp, _Compare, _Allocator > & __x) [inline,`  
`inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**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 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.245 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference 1429**

---

**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.

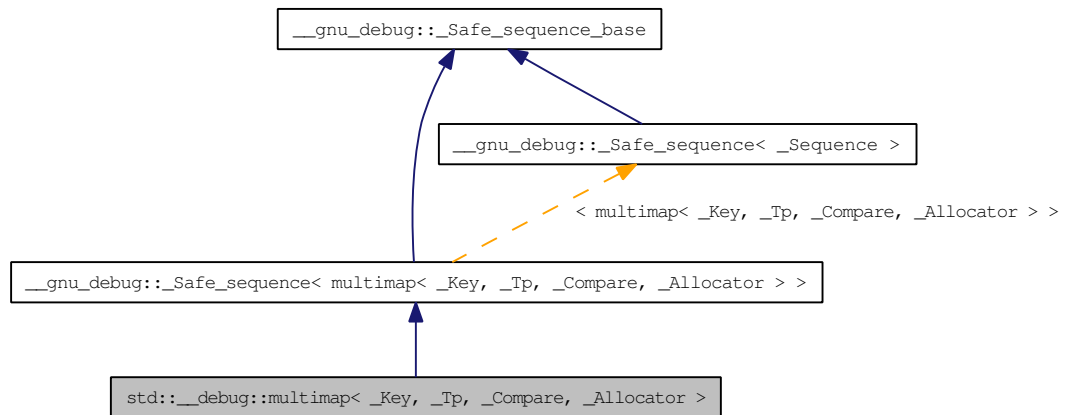
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/map.h](#)

## 5.246 `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`< typename `_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`< typename `_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** (`initializer_list< value_type > __l`, `const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- **multimap** (`multimap &&__x`)
- **multimap** (`const _Base &__x`)
- **multimap** (`const multimap &__x`)
- `template<typename _InputIterator >`  
**multimap** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multimap** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, multimap< _Key, _Tp, _Compare, _Allocator > > &__x)`
- `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`
- `const_iterator end () const`
- `iterator end ()`
- `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 (iterator __first, iterator __last)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __position)`
- `const_iterator find (const key_type &__x) const`
- `iterator find (const key_type &__x)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, const value_type &__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const value_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `multimap & operator= (initializer_list< value_type > __l)`

- `multimap` & `operator=` (`multimap` &&\_\_x)
- `multimap` & `operator=` (const `multimap` &\_\_x)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `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)

## 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 _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`.

### 5.246.2 Member Function Documentation

**5.246.2.1** void `__gnu_debug::Safe_sequence_base::_M_detach_all` ()  
**[protected, inherited]**

Detach all iterators, leaving them singular.



**5.246 std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator > Class**  
**Template Reference** **1433**

---

**5.246.2.2 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.

**5.246.2.3 \_\_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_iter()`.

**5.246.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all () const**  
**[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::append()`,  
`__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator+=(())`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::push_back()`.

**5.246.2.5 void \_\_gnu\_debug::Safe\_sequence< multimap< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_invalidate\_if (\_Predicate \_\_pred)**  
**[inline, inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.246.2.6** `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).

**5.246.2.7** `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.

**5.246.2.8** `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp,`  
`_Compare, _Allocator > >::M_transfer_iter (const Safe_iterator<`  
`_Iterator, multimap< _Key, _Tp, _Compare, _Allocator > > & __x)`  
**[inline, inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**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 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.246 std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator > Class**  
**Template Reference** **1435**

---

**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.

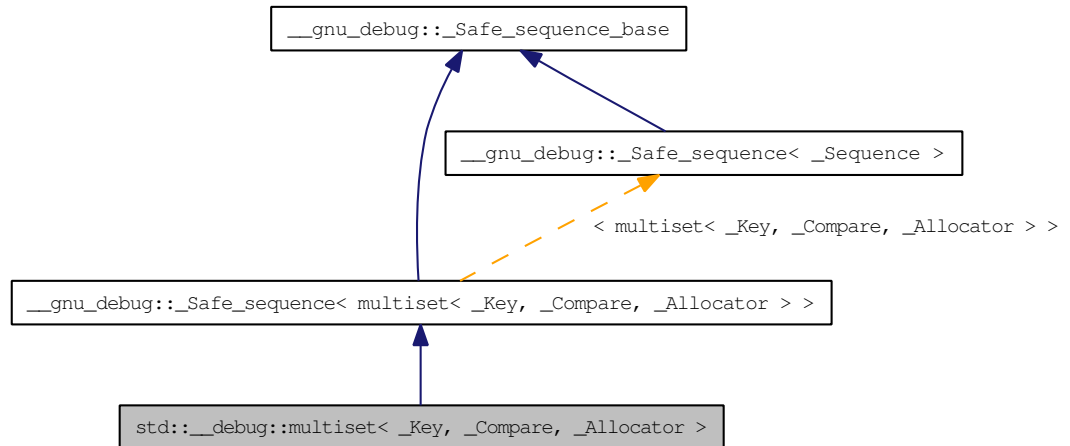
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

## 5.247 `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< typename _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< typename _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** (`initializer_list< value_type > __l`, `const _Compare &__comp=` `_Compare()`, `const allocator_type &__a=allocator_type()`)
- **multiset** (`multiset &&__x`)
- **multiset** (`const _Base &__x`)
- **multiset** (`const multiset &__x`)
- `template<typename _InputIterator >`  
**multiset** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=` `_Compare()`, `const _Allocator &__a=` `_Allocator()`)
- **multiset** (`const _Compare &__comp=` `_Compare()`, `const _Allocator &__a=` `_Allocator()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, multiset< _Key, _Compare, _Allocator > > &__x)`
- `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`
- `const_iterator end () const`
- `iterator end ()`
- `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 (iterator __first, iterator __last)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __position)`
- `const_iterator find (const key_type &__x) const`
- `iterator find (const key_type &__x)`
- `void insert (initializer_list< value_type > __l)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, const value_type &__x)`
- `iterator insert (const value_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `multiset & operator= (initializer_list< value_type > __l)`

- `multiset & operator= (multiset &&__x)`
- `multiset & operator= (const multiset &__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `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)`

## 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 _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`.

### 5.247.2 Member Function Documentation

**5.247.2.1** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()`  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.247 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator > Class Template Reference** **1439**

---

**5.247.2.2 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.

**5.247.2.3 \_\_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_iter()`.

**5.247.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all () const**  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::append()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator+=(())`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::push_back()`.

**5.247.2.5 void \_\_gnu\_debug::Safe\_sequence< multiset< \_Key, \_Compare, \_Allocator > >::\_M\_invalidate\_if (\_Predicate \_\_pred)** [inline, inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.247.2.6** `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).

**5.247.2.7** `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.

**5.247.2.8** `void __gnu_debug::Safe_sequence< multiset< _Key, _Compare,`  
`_Allocator > >::M_transfer_iter (const Safe_iterator< _Iterator,`  
`multiset< _Key, _Compare, _Allocator > & __x) [inline,`  
`inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.247.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_`  
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.



**5.247 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator > Class Template Reference 1441**

---

**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.

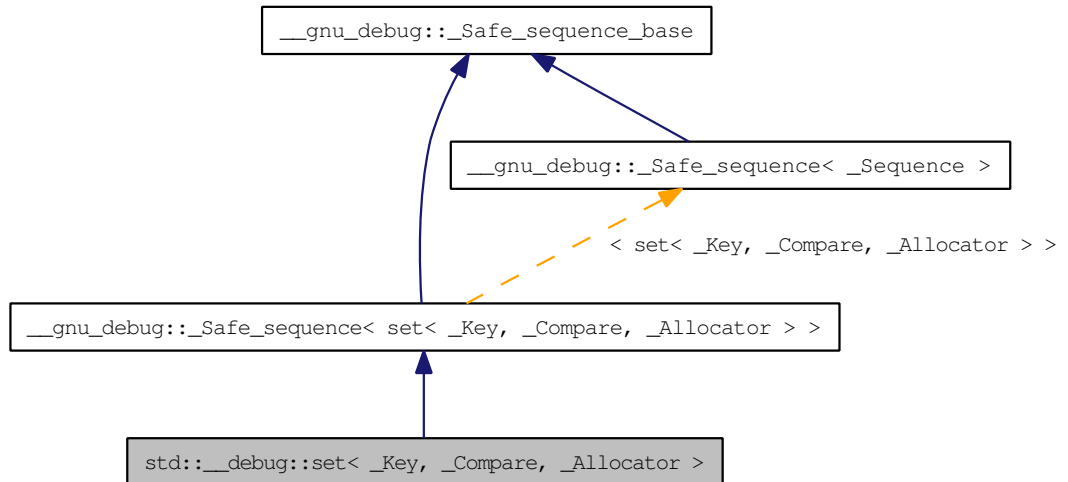
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

## 5.248 `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< typename _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< typename _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(initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `set(set &&__x)`
- `set(const _Base &__x)`
- `set(const set &__x)`
- `template<typename _InputIterator > set(_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())`
- `set(const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())`
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter(const _Safe_iterator< _Iterator, set< _Key, _Compare, _Allocator > > &__x)`
- `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`
- `const_iterator end () const`
- `iterator end ()`
- `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(iterator __first, iterator __last)`
- `size_type erase(const key_type &__x)`
- `iterator erase(iterator __position)`
- `const_iterator find(const key_type &__x) const`
- `iterator find(const key_type &__x)`
- `void insert(initializer_list< value_type > __l)`
- `template<typename _InputIterator > void insert(_InputIterator __first, _InputIterator __last)`
- `iterator insert(iterator __position, const value_type &__x)`
- `std::pair< iterator, bool > insert(const value_type &__x)`
- `const_iterator lower_bound(const key_type &__x) const`
- `iterator lower_bound(const key_type &__x)`
- `set & operator=(initializer_list< value_type > __l)`

- `set & operator= (set &&__x)`
- `set & operator= (const set &__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `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)`

## 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 _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`.

### 5.248.2 Member Function Documentation

**5.248.2.1** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()`  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.248 std::\_\_debug::set< \_Key, \_Compare, \_Allocator > Class Template Reference**

1445

**5.248.2.2 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.

**5.248.2.3 \_\_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_iter()`.

**5.248.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all () const [inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::append()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator+=(())`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::push_back()`.

**5.248.2.5 void \_\_gnu\_debug::Safe\_sequence< set< \_Key, \_Compare, \_Allocator > >::\_M\_invalidate\_if (\_Predicate \_\_pred) [inline, inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.248.2.6** `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).

**5.248.2.7** `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.

**5.248.2.8** `void __gnu_debug::Safe_sequence< set< _Key, _Compare,`  
`_Allocator > >::M_transfer_iter (const Safe_iterator< _Iterator,`  
`set< _Key, _Compare, _Allocator > & __x) [inline,`  
`inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.248.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_`  
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.248 std::\_\_debug::set<\_Key, \_Compare, \_Allocator > Class Template Reference** **1447**

---

**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.

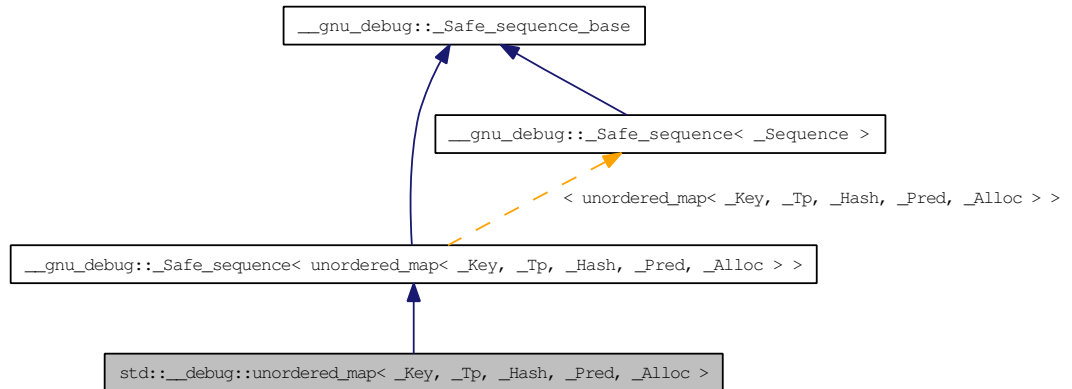
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/set.h](#)

## 5.249 `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< typename _Base::const_iterator, unordered_map >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::__Safe_iterator< typename _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** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_map** (`unordered_map &&__x`)
- **unordered\_map** (`const _Base &__x`)
- **unordered\_map** (`const unordered_map &__x`)



## 5.249 `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference 1449

---

- `template<typename _InputIterator >`  
`unordered_map` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_map` (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `const _Base & _M_base` () const
- `_Base & _M_base` ()
- `void _M_invalidate_all` () const
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_transfer_iter` (`const _Safe_iterator< _Iterator, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > > &__x`)
- `const_iterator begin` () const
- `iterator begin` ()
- `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`)
- `const_iterator find` (`const key_type &__key`) const
- `iterator find` (`const key_type &__key`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `iterator insert` (`const_iterator`, `const value_type &__obj`)
- `std::pair< iterator, bool > insert` (`const value_type &__obj`)
- `unordered_map & operator=` (`initializer_list< value_type > __l`)
- `unordered_map & operator=` (`unordered_map &&__x`)
- `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)

### 5.249.1 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 52 of file [debug/unordered\\_map](#).

### 5.249.2 Member Function Documentation

**5.249.2.1** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_all](#) ()  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.249.2.2** 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](#).

**5.249.2.3** [\\_\\_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\\_iter](#)().

**5.249 std::\_\_debug::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class  
Template Reference 1451**

---

**5.249.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all () const  
[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::push_back()`.

**5.249.2.5 void \_\_gnu\_debug::Safe\_sequence<unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >>::\_M\_invalidate\_if (\_Predicate \_\_pred)  
[inline, inherited]**

Invalidates all iterators  $x$  that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.249.2.6 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).

**5.249.2.7 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.

**5.249.2.8** `void __gnu_debug::_Safe_sequence< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_transfer_iter (const _Safe_iterator< _Iterator, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > > & __x) [inline, inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.249.3.2** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**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 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.250 `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< typename _Base::const_iterator, unordered_multimap >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< typename _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** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multimap** (`unordered_multimap &&__x`)
- **unordered\_multimap** (`const _Base &__x`)
- **unordered\_multimap** (`const unordered_multimap &__x`)
- `template<typename _InputIterator >`  
**unordered\_multimap** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multimap** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`

- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Alloc` > > &\_\_x)
- `const_iterator` `begin` () const
- `iterator` `begin` ()
- `const_iterator` `cbegin` () const
- `const_iterator` `end` () 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)
- `const_iterator` `find` (const `key_type` &\_\_key) const
- `iterator` `find` (const `key_type` &\_\_key)
- template<typename `_InputIterator` >  
void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void `insert` (`std::initializer_list`< `value_type` > \_\_l)
- `iterator` `insert` (`const_iterator`, const `value_type` &\_\_obj)
- `iterator` `insert` (const `value_type` &\_\_obj)
- `unordered_multimap` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `unordered_multimap` & `operator=` (`unordered_multimap` &&\_\_x)
- `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)

### 5.250.1 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 296 of file `debug/unordered_map`.

### 5.250.2 Member Function Documentation

**5.250.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.250.2.2** `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`.

**5.250.2.3** `__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_iter()`.

**5.250.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

**5.250.2.5** `void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_invalidate_if ( _Predicate __pred)`  
[inline, inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.250.2.6** `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).

**5.250.2.7** `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.

**5.250.2.8** `void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_transfer_iter (const`  
`Safe_iterator< Iterator, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > > & __x)` [inline, inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.



### 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 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

#### 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 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

#### 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 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.251 `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`< typename `_Base::const_iterator`, `unordered_multiset` > **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator`< typename `_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` (`initializer_list`< `value_type` > `__l`, `size_type` `__n`=10, `const hasher` &`__hf`=`hasher`(), `const key_equal` &`__eq`=`key_equal`(), `const allocator_type` &`__a`=`allocator_type`())
- `unordered_multiset` (`unordered_multiset` &&`__x`)
- `unordered_multiset` (`const _Base` &`__x`)
- `unordered_multiset` (`const unordered_multiset` &`__x`)
- `template`<typename `_InputIterator` >  
`unordered_multiset` (`_InputIterator` `__f`, `_InputIterator` `__l`, `size_type` `__n`=10, `const hasher` &`__hf`=`hasher`(), `const key_equal` &`__eq`=`key_equal`(), `const allocator_type` &`__a`=`allocator_type`())
- `unordered_multiset` (`size_type` `__n`=10, `const hasher` &`__hf`=`hasher`(), `const key_equal` &`__eq`=`key_equal`(), `const allocator_type` &`__a`=`allocator_type`())
- `const _Base` & `_M_base` () `const`
- `_Base` & `_M_base` ()
- `void` `_M_invalidate_all` () `const`
- `void` `_M_invalidate_if` (`_Predicate` `__pred`)
- `void` `_M_transfer_iter` (`const` `Safe_iterator`< `_Iterator`, `unordered_multiset`< `_Value`, `_Hash`, `_Pred`, `_Alloc` > > &`__x`)

- `const_iterator begin ()` const
- `iterator begin ()`
- `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)`
- `const_iterator find (const key_type &__key)` const
- `iterator find (const key_type &__key)`
- `template<typename _InputIterator > void insert (_InputIterator __first, _InputIterator __last)`
- void `insert (std::initializer_list< value_type > __l)`
- `iterator insert (const_iterator, const value_type &__obj)`
- `iterator insert (const value_type &__obj)`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `unordered_multiset & operator= (unordered_multiset &&__x)`
- `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)`

### 5.251.1 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 295 of file `debug/unordered_set`.

### 5.251.2 Member Function Documentation

**5.251.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.251.2.2** `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`.

**5.251.2.3** `__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_iter()`.

**5.251.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`  
**[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::append()`,  
`__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::assign()`, `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >::c_str()`, `__gnu_debug::basic_`

## 5.251 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc >` Class Template Reference 1461

---

`string<_CharT, _Traits, _Allocator >::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >::push_back()`.

### 5.251.2.5 `void __gnu_debug::Safe_sequence< unordered_multiset<_Value, _Hash, _Pred, _Alloc > >::M_invalidate_if (_Predicate __pred)` [`inline`, `inherited`]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

### 5.251.2.6 `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).

### 5.251.2.7 `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.

### 5.251.2.8 `void __gnu_debug::Safe_sequence< unordered_multiset<_Value, _Hash, _Pred, _Alloc > >::M_transfer_iter (const _Safe_iterator<_Iterator, unordered_multiset<_Value, _Hash, _Pred, _Alloc > > & __x)` [`inline`, `inherited`]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

## 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 166 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 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 163 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 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.

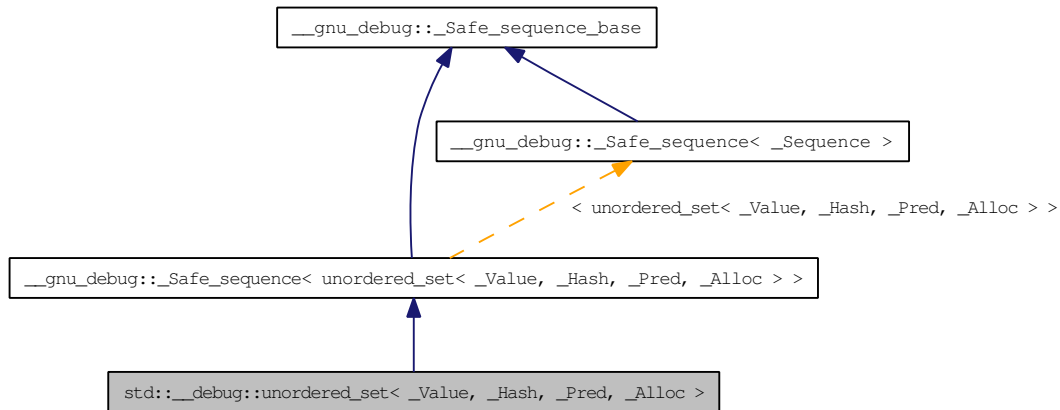
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.252 `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< typename _Base::const_iterator, unordered_set >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< typename _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** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_set** (`unordered_set &&__x`)
- **unordered\_set** (`const _Base &__x`)
- **unordered\_set** (`const unordered_set &__x`)

- `template<typename _InputIterator >`  
**unordered\_set** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_set** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, unordered_set< _Value, _Hash, _Pred, _Alloc > > &__x)`
- `const_iterator begin () const`
- `iterator begin ()`
- `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)`
- `const_iterator find (const key_type &__key) const`
- `iterator find (const key_type &__key)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const_iterator, const value_type &__obj)`
- `std::pair< iterator, bool > insert (const value_type &__obj)`
- `unordered_set & operator= (initializer_list< value_type > __l)`
- `unordered_set & operator= (unordered_set &&__x)`
- `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)`

### 5.252.1 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 52 of file `debug/unordered_set`.

### 5.252.2 Member Function Documentation

**5.252.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.252.2.2** `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`.

**5.252.2.3** `__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_iter()`.

**5.252.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`  
**[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

Referenced by `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::append()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::assign()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::c_str()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::clear()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::data()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator+=()`, `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::operator=()`, and `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>::push_back()`.

**5.252.2.5** `void __gnu_debug::Safe_sequence<unordered_set<_Value, _Hash, _Pred, _Alloc>>::_M_invalidate_if (_Predicate _pred)`  
**[inline, inherited]**

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.252.2.6** `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).

**5.252.2.7** `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 std::\_\_debug::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference 1467

---

**5.252.2.8** void \_\_gnu\_debug::Safe\_sequence< unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc >>::M\_transfer\_iter(const Safe\_iterator<\_Iterator, unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc >> & \_\_x)  
[inline, inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 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 166 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_invalidate\_if(), \_\_gnu\_debug::Safe\_iterator<\_Iterator, \_Sequence >::M\_invalidate\_single(), and \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_iter().

**5.252.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_invalidate\_if(), \_\_gnu\_debug::Safe\_iterator<\_Iterator, \_Sequence >::M\_invalidate\_single(), and \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_iter().

**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.

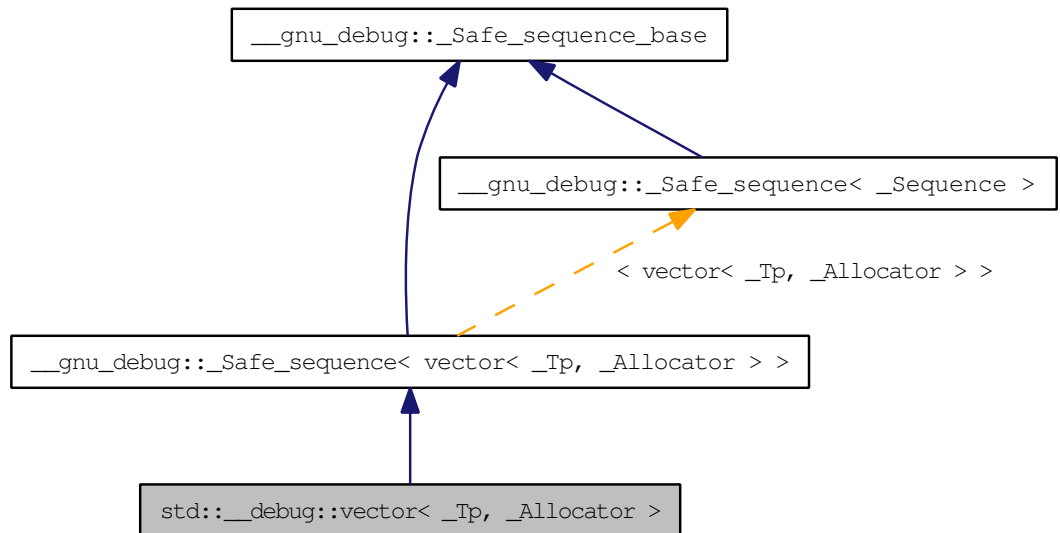
Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.253 `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< typename _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< typename _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** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **vector** ([vector](#) &&\_\_x)
- **vector** (const [\\_Base](#) &\_\_x)
- **vector** (const [vector](#) &\_\_x)
- template<class [\\_InputIterator](#) >  
**vector** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **vector** (size\_type \_\_n, const [\\_Tp](#) &\_\_value=[\\_Tp](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **vector** (const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [\\_Base](#) & [\\_M\\_base](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [vector](#)< [\\_Tp](#), [\\_Allocator](#) > > &\_\_x)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- void **assign** (size\_type \_\_n, const [\\_Tp](#) &\_\_u)
- template<typename [\\_InputIterator](#) >  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- const\_reference **back** () const
- reference **back** ()
- 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
- template<typename... [\\_Args](#) >  
[iterator](#) **emplace** ([iterator](#) \_\_position, [\\_Args](#) &&...\_\_args)
- template<typename... [\\_Args](#) >  
void **emplace\_back** ([\\_Args](#) &&...\_\_args)
- const\_iterator **end** () const
- iterator **end** ()
- iterator **erase** ([iterator](#) \_\_first, [iterator](#) \_\_last)
- iterator **erase** ([iterator](#) \_\_position)
- const\_reference **front** () const
- reference **front** ()

- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `void insert (iterator __position, initializer_list< value_type > __l)`
- `template<typename _Up = _Tp>`  
`__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, iterator >::__type insert (iterator __position, _Tp &&__x)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `vector & operator= (initializer_list< value_type > __l)`
- `vector & operator= (vector &&__x)`
- `vector & operator= (const vector &__x)`
- `const_reference operator[] (size_type __n) const`
- `reference operator[] (size_type __n)`
- `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, _Tp __c=_Tp())`
- `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)`

### 5.253.1 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`.

### 5.253.2 Constructor & Destructor Documentation

```
5.253.2.1 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 94 of file `debug/vector`.

### 5.253.3 Member Function Documentation

```
5.253.3.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ()
[protected, inherited]
```

Detach all iterators, leaving them singular.

```
5.253.3.2 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`.

```
5.253.3.3 __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_iter()`.

**5.253.3.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all () const [inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

Referenced by \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::append(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::assign(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::c\_str(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::clear(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::data(), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::operator+=( ), \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::operator=( ), and \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >::push\_back().

**5.253.3.5 void \_\_gnu\_debug::Safe\_sequence< vector< \_Tp, \_Allocator > >::M\_invalidate\_if(\_Predicate \_\_pred) [inline, inherited]**

Invalidates all iterators  $x$  that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.253.3.6 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).

**5.253.3.7 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.

**5.253.3.8 void \_\_gnu\_debug::Safe\_sequence< vector< \_Tp, \_Allocator > >::M\_transfer\_iter (const \_Safe\_iterator< \_Iterator, vector< \_Tp, \_Allocator > > & \_\_x) [inline, inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.



## 5.253.4 Member Data Documentation

### 5.253.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

### 5.253.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

### 5.253.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

## 5.254 `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`

#### 5.254.1 Detailed Description

`template<typename _Tp> struct std::__declval_protector< _Tp >`

`declval`

Definition at line 616 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.255 `std::__exception_ptr::exception_ptr` Class Reference

An opaque pointer to an arbitrary [exception](#).

### Public Types

- `typedef void(exception_ptr::* __safe_bool )()`

### Public Member Functions

- `exception_ptr (exception_ptr &&__o) throw ()`
- `exception_ptr (const exception_ptr &) throw ()`
- `exception_ptr (__safe_bool) throw ()`
- `const type_info * __cxa_exception_type () const __attribute__((__pure__)) throw ()`
- `exception_ptr & operator= (exception_ptr &&__o) throw ()`
- `exception_ptr & operator= (const exception_ptr &) 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)`

#### 5.255.1 Detailed Description

An opaque pointer to an arbitrary [exception](#).

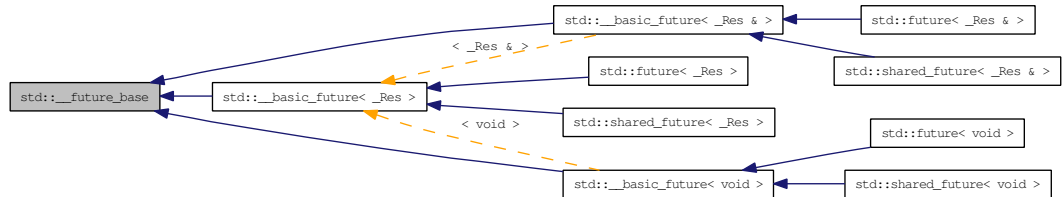
Definition at line 73 of file `exception_ptr.h`.

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 5.256 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\\_base](#)  
*Base class for results.*
- class [\\_State](#)  
*Shared state between a [promise](#) and one or more associated futures.*

### 5.256.1 Detailed Description

Base class and enclosing scope.

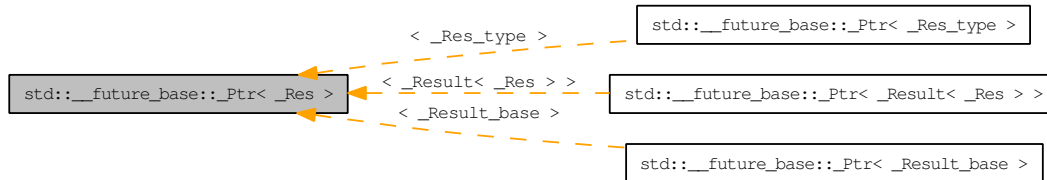
Definition at line 136 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.257 `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**

#### 5.257.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Ptr<_Res>`

A `unique_ptr` based on the instantiating type.

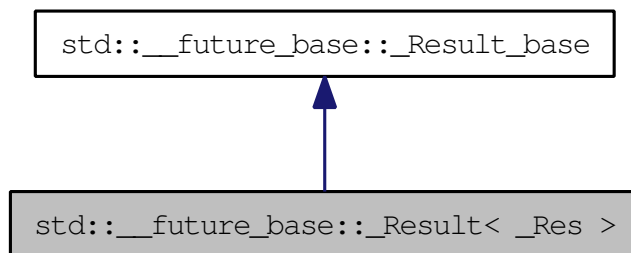
Definition at line 211 of file `future`.

The documentation for this struct was generated from the following file:

- `future`

## 5.258 `std::__future_base::_Result< _Res >` Struct Template Reference

Result. Inheritance diagram for `std::__future_base::_Result< _Res >`:



### Public Member Functions

- `void _M_set (_Res &&__res)`
- `void _M_set (const _Res &__res)`
- `_Res & _M_value ()`

### Public Attributes

- `exception_ptr _M_error`

### 5.258.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result< _Res >`

Result.

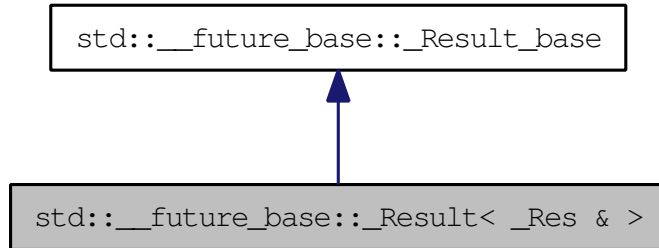
Definition at line 161 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.259 `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`

### 5.259.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result<_Res &>`

Partial specialization for reference types.

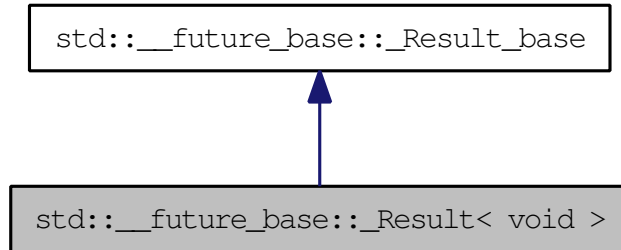
Definition at line 455 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.260 `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`

### 5.260.1 Detailed Description

`template<> struct std::__future_base::_Result< void >`

Explicit specialization for void.

Definition at line 471 of file future.

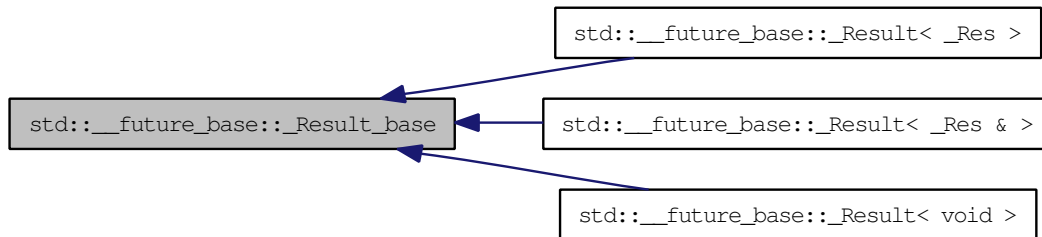
The documentation for this struct was generated from the following file:

- [future](#)



## 5.261 `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`

#### 5.261.1 Detailed Description

Base class for results.

Definition at line 139 of file `future`.

The documentation for this struct was generated from the following file:

- `future`

## 5.262 `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

- `static _Setter< void, void > __setter` ([promise](#)< `void` > \*\_\_prom)
- `template<typename _Res > static _Setter< _Res, __exception_ptr_tag > __setter` (`exception_ptr &__ex`, [promise](#)< `_Res` > \*\_\_prom)
- `template<typename _Res, typename _Arg > static _Setter< _Res, _Arg && > __setter` ([promise](#)< `_Res` > \*\_\_prom, `_Arg &&__arg`)
- `template<typename _Tp > static bool _S_check` (const [shared\\_ptr](#)< `_Tp` > &\_\_p)

#### 5.262.1 Detailed Description

Shared state between a [promise](#) and one or more associated futures.

Definition at line 262 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.263 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.263.1 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.

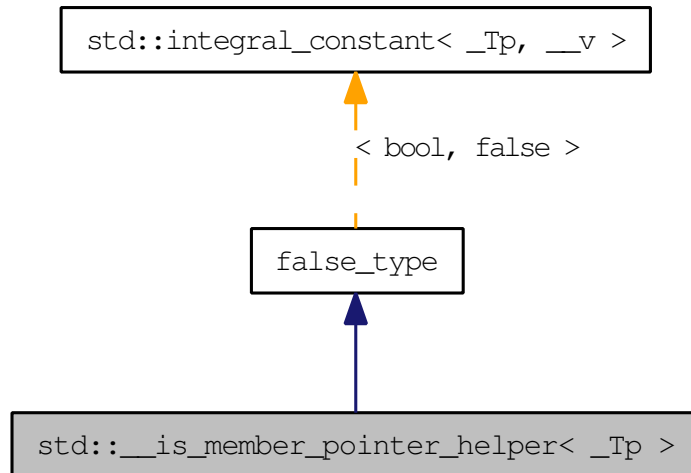
Definition at line 1420 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.264 `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< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.264.1 Detailed Description

```
template<typename _Tp> struct std::__is_member_pointer_helper< _Tp >
```

`is_member_pointer`

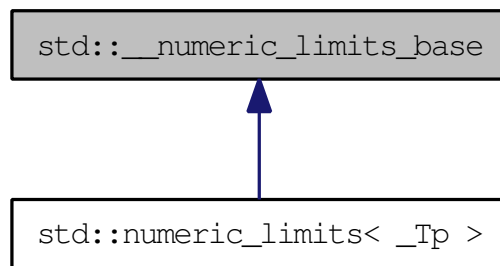
Definition at line 295 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.265 std::\_\_numeric\_limits\_base Struct Reference

Part of `std::numeric_limits`. Inheritance diagram for `std::__numeric_limits_base`:



### Static Public Attributes

- static const int `digits`
- static const int `digits10`
- static const `float_denorm_style` `has_denorm`
- static const bool `has_denorm_loss`
- static const bool `has_infinity`
- static const bool `has_quiet_NaN`
- static const bool `has_signaling_NaN`
- static const bool `is_bounded`
- static const bool `is_exact`
- static const bool `is_iec559`
- static const bool `is_integer`
- static const bool `is_modulo`
- static const bool `is_signed`
- static const bool `is_specialized`
- static const int `max_digits10`
- static const int `max_exponent`
- static const int `max_exponent10`
- static const int `min_exponent`
- static const int `min_exponent10`
- static const int `radix`
- static const `float_round_style` `round_style`
- static const bool `tinyness_before`
- static const bool `traps`

### 5.265.1 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 190 of file `limits`.

### 5.265.2 Member Data Documentation

#### 5.265.2.1 `const 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 199 of file `limits`.

#### 5.265.2.2 `const int std::__numeric_limits_base::digits10` [static]

The number of base 10 digits that can be represented without change.

Definition at line 201 of file `limits`.

#### 5.265.2.3 `const float_denorm_style std::__numeric_limits_base::has_denorm` [static]

See [std::float\\_denorm\\_style](#) for more information.

Definition at line 244 of file `limits`.

#### 5.265.2.4 `const 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 247 of file `limits`.

**5.265.2.5 const bool std::\_\_numeric\_limits\_base::has\_infinity [static]**

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

**5.265.2.6 const 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 239 of file limits.

**5.265.2.7 const 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 242 of file limits.

**5.265.2.8 const 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 255 of file limits.

**5.265.2.9 const 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 216 of file limits.

**5.265.2.10 const 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 251 of file limits.

**5.265.2.11 const 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 211 of file limits.

#### **5.265.2.12** `const 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 260 of file limits.

#### **5.265.2.13** `const bool std::__numeric_limits_base::is_signed` `[static]`

True if the type is signed.

Definition at line 208 of file limits.

#### **5.265.2.14** `const 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 194 of file limits.

#### **5.265.2.15** `const 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 205 of file limits.

#### **5.265.2.16** `const 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 230 of file limits.

#### **5.265.2.17** `const 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 233 of file limits.



**5.265.2.18 const 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 223 of file `limits`.

**5.265.2.19 const 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 226 of file `limits`.

**5.265.2.20 const 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 219 of file `limits`.

**5.265.2.21 const 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 269 of file `limits`.

**5.265.2.22 const bool std::\_\_numeric\_limits\_base::tinyness\_before [static]**

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file `limits`.

**5.265.2.23 const bool std::\_\_numeric\_limits\_base::traps [static]**

True if trapping is implemented for this type.

Definition at line 263 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.266 `std::__parallel::_CRandNumber< _MustBeInt >` Struct Template Reference

Functor wrapper for `std::rand()`.

### Public Member Functions

- `int operator() (int __limit)`

#### 5.266.1 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](#)

## 5.267 `std::__profile::bitset<_Nb>` Class Template Reference

Class `std::bitset` wrapper with performance instrumentation.

Inherits `bitset<_Nb>`.

### Public Member Functions

- `bitset` (const char \*\_\_str)
- `bitset` (const `_Base` & \_\_x)
- `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, type-  
name `std::basic_string<_CharT, _Traits, _Alloc >::size_type` \_\_n, `_CharT` \_\_zero,  
`_CharT` \_\_one=`_CharT('1')`)
- `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`))
- `bitset` (unsigned long long \_\_val)
- const `_Base` & `_M_base` () const
- `_Base` & `_M_base` ()
- `bitset<_Nb>` & `flip` (size\_t \_\_pos)
- `bitset<_Nb>` & `flip` ()
- 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)
- bool `operator[]` (size\_t \_\_pos) const
- reference `operator[]` (size\_t \_\_pos)
- `bitset<_Nb>` & `operator^=` (const `bitset<_Nb>` & \_\_rhs)
- `bitset<_Nb>` & `operator|=` (const `bitset<_Nb>` & \_\_rhs)
- `bitset<_Nb>` `operator~` () const
- `bitset<_Nb>` & `reset` (size\_t \_\_pos)
- `bitset<_Nb>` & `reset` ()
- `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
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **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<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **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<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** () const

### 5.267.1 Detailed Description

template<size\_t \_Nb> class [std::\\_\\_profile::bitset](#)< \_Nb >

Class [std::bitset](#) wrapper with performance instrumentation.

Definition at line 40 of file profile/bitset.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

## 5.268 `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** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **deque** ([deque](#) &&\_\_x)
- **deque** (const [\\_Base](#) &\_\_x)
- **deque** (const [deque](#) &\_\_x)
- template<class [\\_InputIterator](#) >  
**deque** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** (size\_type \_\_n, const [\\_Tp](#) &\_\_value=[\\_Tp](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** (const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [\\_Base](#) & [\\_M\\_base](#) ()
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- void **assign** (size\_type \_\_n, const [\\_Tp](#) &\_\_t)
- template<class [\\_InputIterator](#) >  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- const\_reference **back** () const

- reference **back** ()
- const\_iterator **begin** () const
- iterator **begin** ()
- const\_iterator **cbegin** () const
- const\_iterator **kend** () 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)
- const\_iterator **end** () const
- iterator **end** ()
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- const\_reference **front** () const
- reference **front** ()
- template<class \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- void **insert** (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- iterator **insert** (iterator \_\_position, \_Tp &&\_\_x)
- iterator **insert** (iterator \_\_position, const \_Tp &\_\_x)
- deque & **operator=** (initializer\_list< value\_type > \_\_l)
- deque & **operator=** (deque &&\_\_x)
- deque & **operator=** (const deque &\_\_x)
- const\_reference **operator[]** (size\_type \_\_n) const
- reference **operator[]** (size\_type \_\_n)
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_Tp &&\_\_x)
- void **push\_back** (const \_Tp &\_\_x)
- void **push\_front** (\_Tp &&\_\_x)
- void **push\_front** (const \_Tp &\_\_x)
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rend** () const
- reverse\_iterator **rend** ()
- void **resize** (size\_type \_\_sz, \_Tp \_\_c=\_Tp())
- void **swap** (deque &\_\_x)

### **5.268.1 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](#)

## 5.269 `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** (`initializer_list< value_type > __l`, `const allocator_type &__a=allocator_type()`)
- **list** (`list &&__x`)
- **list** (`const _Base &__x`)
- **list** (`const list &__x`)
- `template<class _InputIterator >`  
**list** (`_InputIterator __first`, `_InputIterator __last`, `const _Allocator &__a=_Allocator()`)
- **list** (`size_type __n`, `const _Tp &__value=_Tp()`, `const _Allocator &__a=_Allocator()`)
- **list** (`const _Allocator &__a=_Allocator()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void _M_profile_find () const`
- `void _M_profile_iterate (int __rewind=0) const`
- `void assign (size_type __n, const _Tp &__t)`



- `template<class _InputIterator >`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- `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)`
- `const_iterator end () const`
- `iterator end ()`
- `iterator erase (iterator __position, iterator __last)`
- `iterator erase (iterator __position)`
- `const_reference front () const`
- `reference front ()`
- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `void insert (iterator __position, initializer_list< value_type > __l)`
- `iterator insert (iterator __position, _Tp &&__x)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `template<typename _Compare >`  
`void merge (list &__x, _Compare __comp)`
- `template<class _Compare >`  
`void merge (list &&__x, _Compare __comp)`
- `void merge (list &__x)`
- `void merge (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `list & operator= (list &&__x)`
- `list & operator= (const list &__x)`
- `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, \_Tp \_\_c=\_Tp())
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **sort** ()
- 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, iterator \_\_i)
- void **splice** (iterator \_\_position, [list](#) &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, [list](#) &\_\_x)
- void **splice** (iterator \_\_position, [list](#) &&\_\_x)
- void **swap** ([list](#) &\_\_x)
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_binary\_pred)
- void **unique** ()

### 5.269.1 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](#)

## 5.270 `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** (`initializer_list< value_type > __l`, `const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- **map** (`map &&__x`)
- **map** (`const _Base &__x`)
- **map** (`const map &__x`)
- `template<typename _InputIterator >`  
  **map** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **map** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `const _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\\_iterator](#) **cbegin** () const
- [const\\_iterator](#) **rend** () const
- [void](#) **clear** ()
- [size\\_type](#) **count** (const [key\\_type](#) &\_\_x) const
- [const\\_reverse\\_iterator](#) **crbegin** () const
- [const\\_reverse\\_iterator](#) **crend** () const
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- [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** ([iterator](#) \_\_first, [iterator](#) \_\_last)
- [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
- [iterator](#) **erase** ([iterator](#) \_\_position)
- [const\\_iterator](#) **find** (const [key\\_type](#) &\_\_x) const
- [iterator](#) **find** (const [key\\_type](#) &\_\_x)
- [template](#)<typename [\\_InputIterator](#) >  
void **insert** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [iterator](#) **insert** ([iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- [void](#) **insert** ([std::initializer\\_list](#)< [value\\_type](#) > \_\_list)
- [std::pair](#)< [iterator](#), [bool](#) > **insert** (const [value\\_type](#) &\_\_x)
- [const\\_iterator](#) **lower\_bound** (const [key\\_type](#) &\_\_x) const
- [iterator](#) **lower\_bound** (const [key\\_type](#) &\_\_x)
- [map](#) & **operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [map](#) & **operator=** ([map](#) &&\_\_x)
- [map](#) & **operator=** (const [map](#) &\_\_x)
- [mapped\\_type](#) & **operator**[ ] (const [key\\_type](#) &\_\_k)
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- [reverse\\_iterator](#) **rend** ()
- [void](#) **swap** ([map](#) &\_\_x)
- [const\\_iterator](#) **upper\_bound** (const [key\\_type](#) &\_\_x) const
- [iterator](#) **upper\_bound** (const [key\\_type](#) &\_\_x)

### 5.270.1 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.

**5.270 std::\_\_profile::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference** **1501**

---

Definition at line 47 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

## 5.271 `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** (`initializer_list`< `value_type` > `__l`, `const _Compare` & `__c`=`_Compare()`, `const allocator_type` & `__a`=`allocator_type()`)
- **multimap** (`multimap` && `__x`)
- **multimap** (`const _Base` & `__x`)
- **multimap** (`const multimap` & `__x`)
- `template<typename _InputIterator >`  
**multimap** (`_InputIterator` `__first`, `_InputIterator` `__last`, `const _Compare` & `__comp`=`_Compare()`, `const _Allocator` & `__a`=`_Allocator()`)
- **multimap** (`const _Compare` & `__comp`=`_Compare()`, `const _Allocator` & `__a`=`_Allocator()`)
- `const _Base` & `_M_base` () `const`
- `_Base` & `_M_base` ()
- `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
- `const_iterator` **end** () const
- `iterator` **end** ()
- `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** (`iterator` \_\_first, `iterator` \_\_last)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (`iterator` \_\_position)
- `const_iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **find** (const `key_type` &\_\_x)
- `template<typename _InputIterator >`  
void **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator` **insert** (`iterator` \_\_position, const `value_type` &\_\_x)
- void **insert** (`std::initializer_list< value_type >` \_\_list)
- `iterator` **insert** (const `value_type` &\_\_x)
- `const_iterator` **lower\_bound** (const `key_type` &\_\_x) const
- `iterator` **lower\_bound** (const `key_type` &\_\_x)
- `multimap` & **operator=** (`initializer_list< value_type >` \_\_l)
- `multimap` & **operator=** (`multimap` &&\_\_x)
- `multimap` & **operator=** (const `multimap` &\_\_x)
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `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)

### 5.271.1 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](#)



## 5.272 `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** (`initializer_list< value_type > __l`, `const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- **multiset** (`multiset &&__x`)
- **multiset** (`const _Base &__x`)
- **multiset** (`const multiset &__x`)
- `template<typename _InputIterator >`  
**multiset** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multiset** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `const_iterator begin () const`
- `iterator begin ()`
- `const_iterator cbegin () const`

- `const_iterator` **end** () const
- `void` **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `const_iterator` **end** () const
- `iterator` **end** ()
- `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** (`iterator` \_\_first, `iterator` \_\_last)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (`iterator` \_\_position)
- `const_iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **find** (const `key_type` &\_\_x)
- `void` **insert** (`initializer_list`< `value_type` > \_\_l)
- `template`< `typename` `_InputIterator` >  
`void` **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator` **insert** (`iterator` \_\_position, const `value_type` &\_\_x)
- `iterator` **insert** (const `value_type` &\_\_x)
- `const_iterator` **lower\_bound** (const `key_type` &\_\_x) const
- `iterator` **lower\_bound** (const `key_type` &\_\_x)
- `multiset` & **operator=** (`initializer_list`< `value_type` > \_\_l)
- `multiset` & **operator=** (`multiset` &&\_\_x)
- `multiset` & **operator=** (const `multiset` &\_\_x)
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `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)

### 5.272.1 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](#)

## 5.273 `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** (`initializer_list<value_type > __l`, `const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- **set** (`set &&__x`)
- **set** (`const _Base &__x`)
- **set** (`const set &__x`)
- `template<typename _InputIterator >`  
**set** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **set** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `const _Base &_M_base () const`
- `_Base &_M_base ()`
- `const_iterator begin () const`
- `iterator begin ()`
- `const_iterator cbegin () const`

- `const_iterator` **end** () const
- `void` **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `const_iterator` **end** () const
- `iterator` **end** ()
- `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** (`iterator` \_\_first, `iterator` \_\_last)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (`iterator` \_\_position)
- `const_iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **find** (const `key_type` &\_\_x)
- `void` **insert** (`initializer_list`< `value_type` > \_\_l)
- `template`< `typename` \_InputIterator >  
`void` **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `iterator` **insert** (`iterator` \_\_position, const `value_type` &\_\_x)
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &\_\_x)
- `const_iterator` **lower\_bound** (const `key_type` &\_\_x) const
- `iterator` **lower\_bound** (const `key_type` &\_\_x)
- `set` & **operator=** (`initializer_list`< `value_type` > \_\_l)
- `set` & **operator=** (`set` &&\_\_x)
- `set` & **operator=** (const `set` &\_\_x)
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `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)

### 5.273.1 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](#)

## 5.274 `std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_map` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::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 (initializer\_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_map (unordered\_map &&__x)`
- `unordered_map (const _Base &__x)`
- `template<typename _InputIterator >  
unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_map (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())`
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `void clear ()`
- `void insert (const value_type *__first, const value_type *__last)`
- `template<typename _InputIter >  
void insert (_InputIter __first, _InputIter __last)`
- `iterator insert (const_iterator __iter, const value_type &__v)`

- [std::pair](#)< [iterator](#), bool > **insert** (const value\_type &\_\_obj)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_map](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_map](#) & **operator=** ([unordered\\_map](#) &&\_\_x)
- [unordered\\_map](#) & **operator=** (const [unordered\\_map](#) &\_\_x)
- mapped\_type & **operator[]** (const \_Key &k)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_map](#) &\_\_x)

### 5.274.1 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 59 of file profile/unordered\_map.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

## 5.275 `std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::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` ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multimap` ([unordered\\_multimap](#) &&\_\_x)
- `unordered_multimap` (const \_Base &\_\_x)
- `template<typename _InputIterator >`  
`unordered_multimap` (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multimap` (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `const _Base & M_base () const`
- `_Base & M_base ()`
- `void clear ()`
- `void insert (const value_type * __first, const value_type * __last)`
- `template<typename _InputIter >`  
`void insert (_InputIter __first, _InputIter __last)`

- [iterator insert](#) (const\_iterator \_\_iter, const value\_type &\_\_v)
- [iterator insert](#) (const value\_type &\_\_obj)
- void [insert](#) (std::initializer\_list< value\_type > \_\_l)
- [unordered\\_multimap](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)
- [unordered\\_multimap](#) & [operator=](#) (unordered\_multimap &&\_\_x)
- [unordered\\_multimap](#) & [operator=](#) (const unordered\_multimap &\_\_x)
- void [rehash](#) (size\_type \_\_n)
- void [swap](#) (unordered\_multimap &\_\_x)

### 5.275.1 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 285 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)



## 5.276 **std::\_\_profile::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference**

Unordered\_multiset wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_multiset< _Value, _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\_multiset** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`unordered_multiset &&__x`)
- **unordered\_multiset** (`const _Base &__x`)
- `template<typename _InputIterator >`  
**unordered\_multiset** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- void **clear** ()
- void **insert** (`const value_type *__first`, `const value_type *__last`)
- `template<typename _InputIter >`  
void **insert** (`_InputIter __first`, `_InputIter __last`)
- **iterator insert** (`const_iterator __iter`, `const value_type &__v`)
- **iterator insert** (`const value_type &__obj`)
- void **insert** (`std::initializer_list< value_type > __l`)

- [unordered\\_multiset](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_multiset](#) & **operator=** ([unordered\\_multiset](#) &&\_\_x)
- [unordered\\_multiset](#) & **operator=** (const [unordered\\_multiset](#) &\_\_x)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_multiset](#) &\_\_x)

### 5.276.1 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 275 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 5.277 `std::__profile::unordered_set<_Key, _Hash, _Pred, _Alloc >` Class Template Reference

`Unordered_set` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_set<_Key, _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\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** ([unordered\\_set](#) &&\_\_x)
- **unordered\_set** (const \_Base &\_\_x)
- `template<typename _InputIterator >`  
**unordered\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void **clear** ()
- void **insert** (const value\_type \*\_\_first, const value\_type \*\_\_last)
- `template<typename _InputIter >`  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- **iterator insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- `std::pair< iterator, bool >` **insert** (const value\_type &\_\_obj)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- **unordered\_set & operator=** ([initializer\\_list](#)< value\_type > \_\_l)

- [unordered\\_set](#) & **operator=** ([unordered\\_set](#) &&\_\_x)
- [unordered\\_set](#) & **operator=** (const [unordered\\_set](#) &\_\_x)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_set](#) &\_\_x)

### 5.277.1 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 59 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 5.278 std::\_Base\_bitset< \_Nw > Struct Template Reference

Inheritance diagram for std::\_Base\_bitset< \_Nw >:



### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- **\_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)
- \_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 \_WordT **\_S\_maskbit** (size\_t \_\_pos)
- static size\_t **\_S\_whichbit** (size\_t \_\_pos)
- static size\_t **\_S\_whichbyte** (size\_t \_\_pos)
- static size\_t **\_S\_whichword** (size\_t \_\_pos)

## Public Attributes

- `_WordT _M_w [_Nw]`

### 5.278.1 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 69 of file `bitset`.

### 5.278.2 Member Data Documentation

**5.278.2.1** `template<size_t _Nw> _WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

Definition at line 74 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.279 `std::_Base_bitset< 0 >` Struct Template Reference

### Public Types

- typedef unsigned long `_WordT`

### Public Member Functions

- `_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) const
- `_WordT _M_hiword` () const
- `bool _M_is_any` () const
- `bool _M_is_equal` (const `_Base_bitset< 0 >` &) const

### Static Public Member Functions

- static `_WordT _S_maskbit` (size\_t \_\_pos)
- static `size_t _S_whichbit` (size\_t \_\_pos)
- static `size_t _S_whichbyte` (size\_t \_\_pos)
- static `size_t _S_whichword` (size\_t \_\_pos)

### 5.279.1 Detailed Description

`template<> struct std::_Base_bitset< 0 >`

Base class, specialization for no storage (zero-length bitset).

See documentation for [bitset](#).

Definition at line 511 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)



## 5.280 std::\_Base\_bitset< 1 > Struct Template Reference

### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- **\_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)
- **\_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 \_WordT \_S\_maskbit** (size\_t \_\_pos)
- **static size\_t \_S\_whichbit** (size\_t \_\_pos)
- **static size\_t \_S\_whichbyte** (size\_t \_\_pos)
- **static size\_t \_S\_whichword** (size\_t \_\_pos)

### Public Attributes

- **\_WordT \_M\_w**

### 5.280.1 Detailed Description

`template<> struct std::_Base_bitset< 1 >`

Base class, specialization for a single word.

See documentation for [bitset](#).

Definition at line 367 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.281 `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`

### 5.281.1 Detailed Description

```
template<std::size_t _Num> struct std::_Build_index_tuple<_Num >
```

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

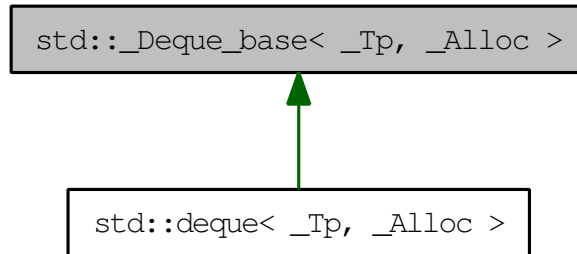
Definition at line 860 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

## 5.282 `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` (`_Deque_base` &&\_\_x)
- `_Deque_base` (const `allocator_type` &\_\_a)
- `_Deque_base` (const `allocator_type` &\_\_a, `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`

### 5.282.1 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 436 of file `stl_deque.h`.

### 5.282.2 Member Function Documentation

**5.282.2.1** `template<typename _Tp, typename _Alloc > void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t num_elements) [inline, protected]`

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 568 of file `stl_deque.h`.

References `std::max()`.

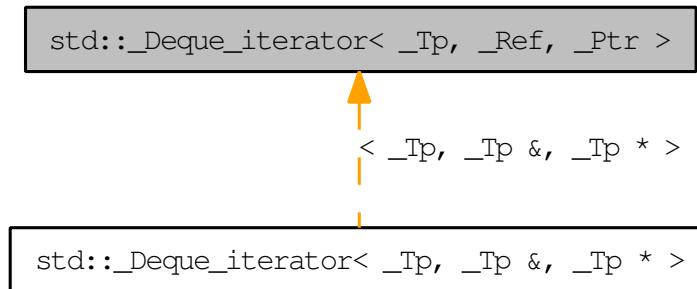
Referenced by `std::deque<_Tp, _Alloc >::_M_range_initialize()`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

## 5.283 `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` (`const iterator &__x`)
- `_Deque_iterator` (`_Tp *__x, _Map_pointer __y`)
- `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--` (`int`)

- [\\_Self](#) & **operator--** ()
- [\\_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**

### 5.283.1 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 103 of file `stl_deque.h`.

### 5.283.2 Member Function Documentation

**5.283.2.1** `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 231 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)



## 5.284 std::\_Derives\_from\_binary\_function< \_Tp > Struct Template Reference

### **5.284 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`

#### **5.284.1 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](#)

## 5.285 `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`

### 5.285.1 Detailed Description

`template<typename _Tp> struct std::_Derives_from_unary_function< _Tp >`

Determines if the type `_Tp` derives from [unary\\_function](#).

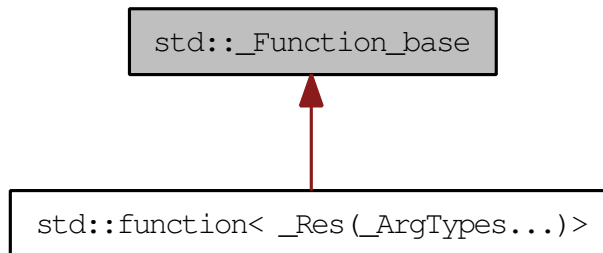
Definition at line 185 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.286 `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`

#### 5.286.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 1498 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

**5.287** `std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct`  
Template Reference 1533

---

## **5.287** `std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct` Template Reference

Turns a function type into a function pointer type.

### **Public Types**

- `typedef _Tp type`

### **5.287.1 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.

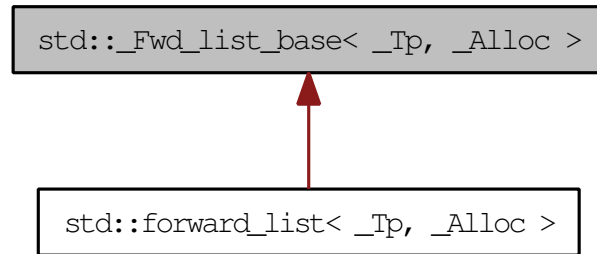
Definition at line 217 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.288 `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, _Tp_alloc_type > _Node`
- typedef `_Fwd_list_node_base< _Tp_alloc_type > _Node_base`
- typedef `_Fwd_list_const_iterator< _Tp, _Tp_alloc_type > const_iterator`
- typedef `_Fwd_list_iterator< _Tp, _Tp_alloc_type > iterator`

### Public Member Functions

- `_Fwd_list_base (_Fwd_list_base &&__lst)`
- `_Fwd_list_base (_Fwd_list_base &&__lst, const _Alloc &__a)`
- `_Fwd_list_base (const _Fwd_list_base &__lst, const _Alloc &__a)`
- `_Fwd_list_base (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, _Tp_alloc_type > >::other _Node_alloc_type`
- typedef `_Alloc::template rebind< _Tp >::other _Tp_alloc_type`

## Protected Member Functions

- `template<typename... _Args> _Node::_Pointer M_create_node (_Args &&... __args)`
- `void M_erase_after (typename _Node_base::_Pointer __pos, typename _Node_base::_Pointer __last)`
- `void M_erase_after (typename _Node_base::_Pointer __pos)`
- `_Node::_Pointer M_get_node ()`
- `template<typename... _Args> _Node_base::_Pointer M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void M_put_node (typename _Node::_Pointer __p)`

## Protected Attributes

- `_Fwd_list_impl M_impl`

### 5.288.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::Fwd_list_base< _Tp, _Alloc >`

Base class for forward\_list.

Definition at line 254 of file forward\_list.h.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.289 `std::_Fwd_list_const_iterator< _Tp, _Alloc >` Struct Template Reference

A `forward_list::const_iterator`.

### Public Types

- `typedef const _Fwd_list_node< _Tp, _Alloc > _Node`
- `typedef const _Fwd_list_node_base< _Alloc > _Node_base`
- `typedef _Fwd_list_const_iterator< _Tp, _Alloc > _Self`
- `typedef _Alloc::difference_type difference_type`
- `typedef _Fwd_list_iterator< _Tp, _Alloc > iterator`
- `typedef std::forward\_iterator\_tag iterator_category`
- `typedef _Alloc::const_pointer pointer`
- `typedef _Alloc::const_reference reference`
- `typedef _Tp value_type`

### Public Member Functions

- `_Fwd_list_const_iterator` (const [iterator](#) &\_\_iter)
- `_Fwd_list_const_iterator` (typename `_Node_base::_Const_pointer` \_\_n)
- `_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

- `_Node_base::_Const_pointer` `_M_node`

#### 5.289.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::_Fwd_list_const_iterator< _Tp, _Alloc >
```

A `forward_list::const_iterator`. All the functions are op overloads.

Definition at line 167 of file `forward_list.h`.

The documentation for this struct was generated from the following file:



**5.289 std::\_Fwd\_list\_const\_iterator<\_Tp, \_Alloc > Struct Template Reference**

- [forward\\_list.h](#)

## 5.290 `std::_Fwd_list_iterator< _Tp, _Alloc >` Struct Template Reference

A `forward_list::iterator`.

### Public Types

- typedef `_Fwd_list_node< _Tp, _Alloc >` `_Node`
- typedef `_Fwd_list_node_base< _Alloc >` `_Node_base`
- typedef `_Fwd_list_iterator< _Tp, _Alloc >` `_Self`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `_Alloc::pointer` `pointer`
- typedef `_Alloc::reference` `reference`
- typedef `_Tp` `value_type`

### Public Member Functions

- `_Fwd_list_iterator` (typename `_Node_base::_Pointer __n`)
- `_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

- `_Node_base::_Pointer _M_node`

#### 5.290.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::_Fwd_list_iterator< _-
Tp, _Alloc >
```

A `forward_list::iterator`. All the functions are op overloads.

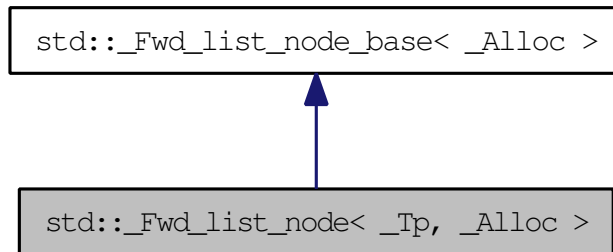
Definition at line 100 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.291 `std::_Fwd_list_node< _Tp, _Alloc >` 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, _Alloc >`:



### Public Types

- typedef `_Alloc::template rebind< \_Fwd\_list\_node\_base< _Alloc >>::other::const_pointer` **\_Const\_pointer**
- typedef `_Alloc::template rebind< \_Fwd\_list\_node< _Tp, _Alloc >>::other::pointer` **\_Pointer**

### Public Member Functions

- `template<typename... _Args> \_Fwd\_list\_node (_Args &&... __args)`
- `void \_M\_reverse\_after ()`
- `void \_M\_transfer\_after (_Pointer __bbegin, _Pointer __bend)`
- `void \_M\_transfer\_after (_Pointer __bbegin)`

### Static Public Member Functions

- `static void swap (\_Fwd\_list\_node\_base &__x, \_Fwd\_list\_node\_base &__y)`

### Public Attributes

- `_Pointer \_M\_next`
- `_Tp \_M\_value`

### 5.291.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::_Fwd_list_node< _Tp,
_Alloc >
```

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.

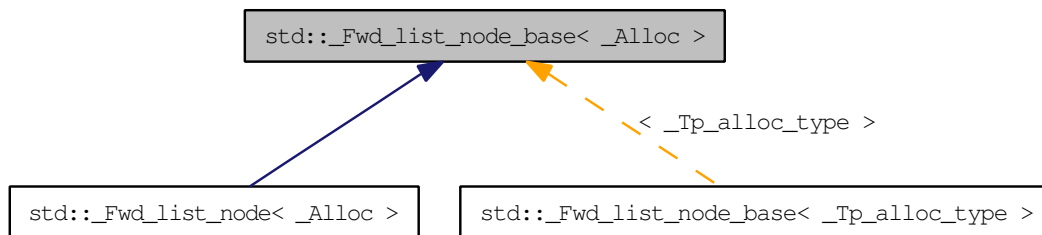
Definition at line 81 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.292 `std::_Fwd_list_node_base<_Alloc>` Struct Template 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<_Alloc>`:



### Public Types

- typedef `_Alloc::template rebind<_Fwd_list_node_base<_Alloc>>::other::const_pointer` **\_Const\_pointer**
- typedef `_Alloc::template rebind<_Fwd_list_node_base<_Alloc>>::other::pointer` **\_Pointer**

### Public Member Functions

- void **\_M\_reverse\_after** ()
- void **\_M\_transfer\_after** (\_Pointer \_\_bbegin, \_Pointer \_\_bend)
- void **\_M\_transfer\_after** (\_Pointer \_\_bbegin)

### Static Public Member Functions

- static void **swap** (\_Fwd\_list\_node\_base &\_\_x, \_Fwd\_list\_node\_base &\_\_y)

### Public Attributes

- `_Pointer` **\_M\_next**

### 5.292.1 Detailed Description

**template<typename \_Alloc> struct std::\_Fwd\_list\_node\_base< \_Alloc >**

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 49 of file `forward_list.h`.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.293 `std::_Has_result_type_helper<_Tp>` Class Template Reference

Inherits `std::_sfnaf_types`.

### Static Public Attributes

- static const bool **value**

### Private Types

- typedef char **\_\_one**

### 5.293.1 Detailed Description

`template<typename _Tp> class std::_Has_result_type_helper<_Tp>`

Actual implementation of `_Has_result_type`, which uses SFINAE to determine if the type `_Tp` has a publicly-accessible member type `result_type`.

Definition at line 72 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.294 `std::_Index_tuple< _Indexes >` Struct Template Reference

### Public Types

- typedef `_Index_tuple< _Indexes..., sizeof...(_Indexes)> __next`

### 5.294.1 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 853 of file `functional`.

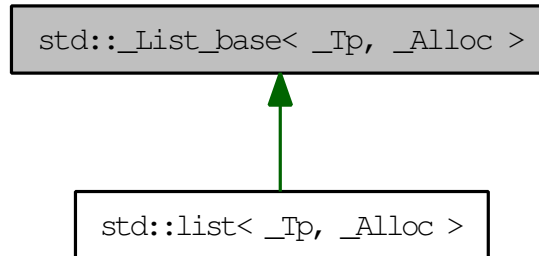
The documentation for this struct was generated from the following file:

- [functional](#)



## 5.295 `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` (`_List_base` &&\_x)
- `_List_base` (const `allocator_type` &\_a)
- 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`

### 5.295.1 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 277 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.296 `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 iterator &__x`)
- `_List_const_iterator` (`const _List_node_base *__x`)
- `bool operator!=(const _Self &__x) const`
- `reference operator* () const`
- `_Self operator++ (int)`
- `_Self & operator++ ()`
- `_Self operator-- (int)`
- `_Self & operator-- ()`
- `pointer operator-> () const`
- `bool operator==(const _Self &__x) const`

### Public Attributes

- `const _List_node_base * _M_node`

#### 5.296.1 Detailed Description

```
template<typename _Tp> struct std::_List_const_iterator< _Tp >
```

A `list::const_iterator`. All the functions are op overloads.

Definition at line 188 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.297 `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` (`_List_node_base *__x`)
- `bool operator!=` (`const _Self &__x`) `const`
- `reference operator*` () `const`
- `_Self operator++` (`int`)
- `_Self & operator++` ()
- `_Self operator--` (`int`)
- `_Self & operator--` ()
- `pointer operator->` () `const`
- `bool operator==` (`const _Self &__x`) `const`

### Public Attributes

- `_List_node_base * _M_node`

#### 5.297.1 Detailed Description

```
template<typename _Tp> struct std::_List_iterator<_Tp>
```

A list::iterator. All the functions are op overloads.

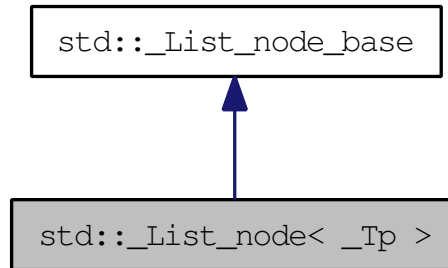
Definition at line 113 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.298 `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`

#### 5.298.1 Detailed Description

```
template<typename _Tp> struct std::_List_node< _Tp >
```

An actual node in the list.

Definition at line 95 of file stl\_list.h.

## **5.298.2 Member Data Documentation**

### **5.298.2.1 template<typename \_Tp> \_Tp std::\_List\_node< \_Tp >::\_M\_data**

< User's data.

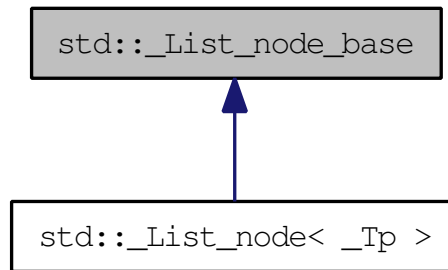
Definition at line 98 of file stl\_list.h.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.299 std::\_List\_node\_base Struct Reference

Common part of a node in the list. Inheritance diagram for std::\_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`

### 5.299.1 Detailed Description

Common part of a node in the list.

Definition at line 71 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)



## 5.300 `std::_Maybe_get_result_type<_Has_result_type, _Funcor > Struct` Template Reference

If we have found a `result_type`, extract it. Inheritance diagram for `std::_Maybe_get_result_type<_Has_result_type, _Funcor >`:



### 5.300.1 Detailed Description

```
template<bool _Has_result_type, typename _Funcor> struct std::_Maybe_get_result_type<_Has_result_type, _Funcor >
```

If we have found a `result_type`, extract it.

Definition at line 96 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.301 `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>` Struct Template Reference

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>`:



#### 5.301.1 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.

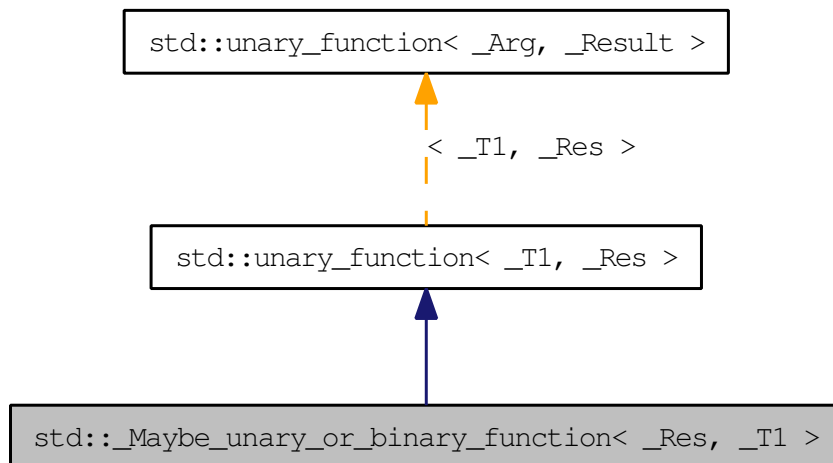
Definition at line 480 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.302 `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`

### 5.302.1 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 484 of file `functional`.

### 5.302.2 Member Typedef Documentation

**5.302.2.1** typedef `_T1` `std::unary_function<_T1, _Res>::argument_type` [inherited]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file stl\_function.h.

**5.302.2.2** `typedef _Res std::unary_function< _T1 , _Res >::result_type`  
`[inherited]`

`result_type` is the return type

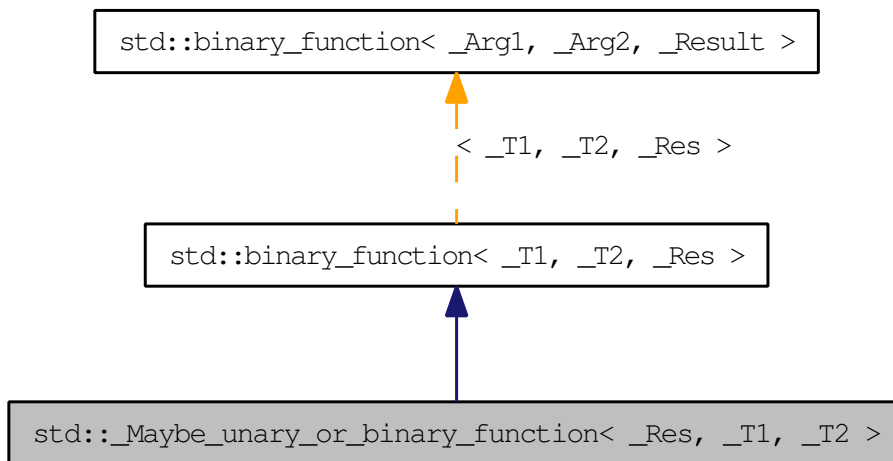
Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.303 `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`

#### 5.303.1 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 489 of file `functional`.

### 5.303.2 Member Typedef Documentation

**5.303.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , _Res  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.303.2.2** `typedef _Res std::binary_function< _T1 , _T2 , _Res >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.303.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , _Res  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.304 `std::_Maybe_wrap_member_pointer<_Tp>` Struct Template Reference 1559

### **5.304 `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`)

#### **5.304.1 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 1048 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.305 `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)`

#### 5.305.1 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 1063 of file `functional`.

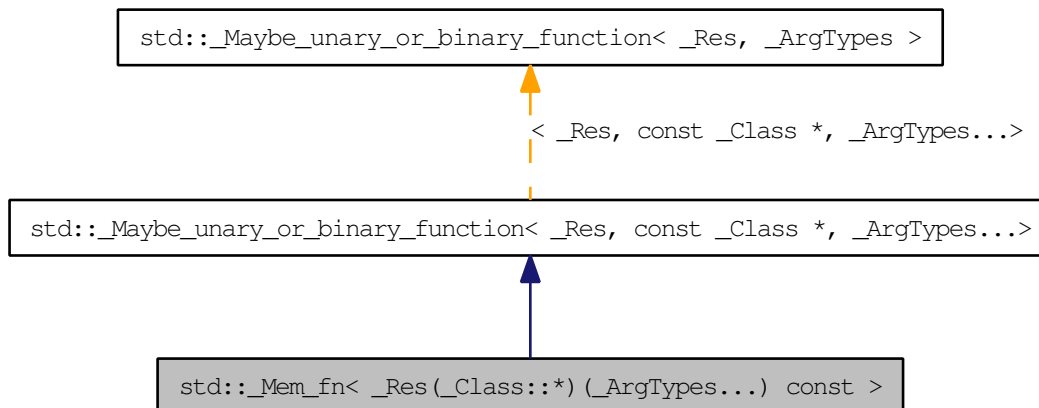
The documentation for this struct was generated from the following file:

- [functional](#)



## 5.306 `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` (`_Function __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`

#### 5.306.1 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 537 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

**5.307 std::\_Mem\_fn< \_Res(\_Class::\*)(\_ArgTypes...) const volatile > Class  
Template Reference 1563**

**5.307 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** (\_Funcor \_\_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

**5.307.1 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.

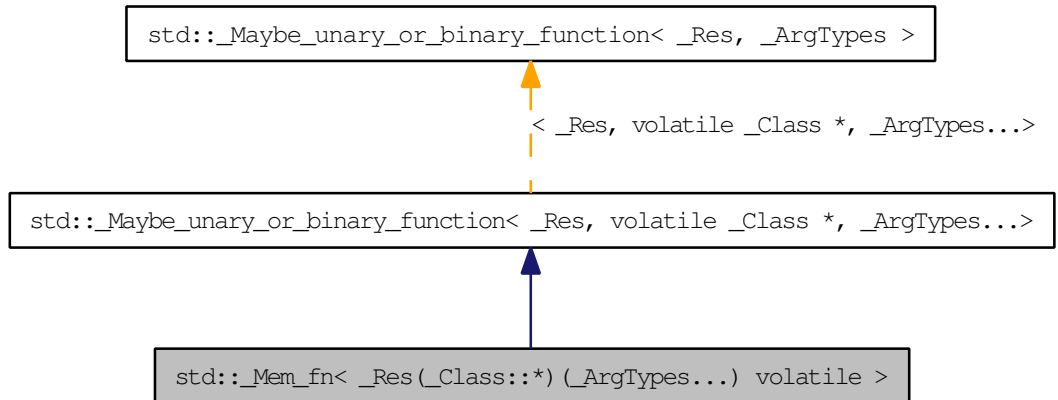
Definition at line 624 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

### 5.308 `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`

#### 5.308.1 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 580 of file `functional`.

The documentation for this class was generated from the following file:

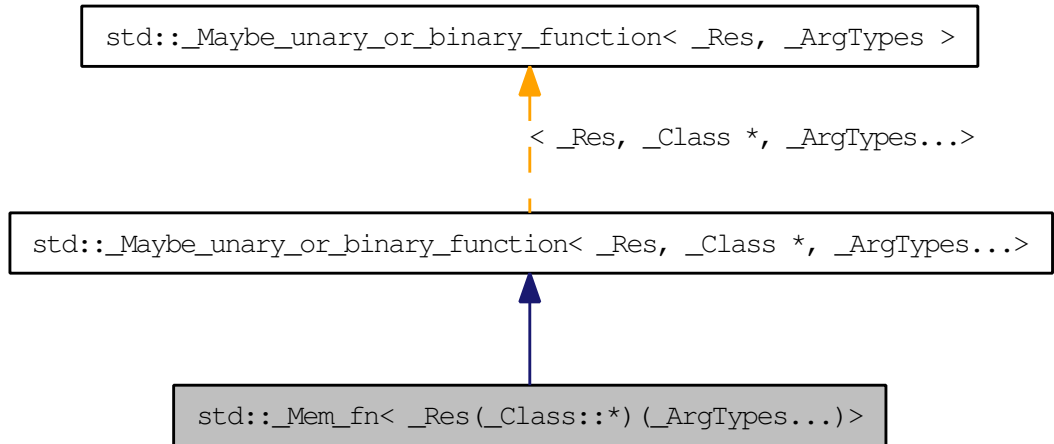
**5.308 std::\_Mem\_fn<\_Res(\_Class::\*)(\_ArgTypes...) volatile > Class Template Reference 1565**

---

- [functional](#)

### 5.309 `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`

#### 5.309.1 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 494 of file `functional`.

### **5.309 std::\_Mem\_fn<\_Res(\_Class::\*)(\_ArgTypes...)> Class Template Reference**

The documentation for this class was generated from the following file:

- [functional](#)

## 5.310 `std::_Mu< _Arg, false, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVArg, typename _Tuple > _CVArg && operator() (_CVArg &&__arg, _Tuple &&)` const volatile

### 5.310.1 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 1024 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)



## 5.311 `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`

### 5.311.1 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 990 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.312 `std::_Mu< _Arg, true, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVArg, typename... _Args> result_of< _CVArg(_Args...)>::type operator() (_CVArg &__arg, tuple< _Args...> &&__tuple) const volatile`

### 5.312.1 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 949 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.313 `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`

#### 5.313.1 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 928 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.314 `std::_Placeholder< _Num >` Struct Template Reference

The type of placeholder objects defined by `libstdc++`.

### 5.314.1 Detailed Description

```
template<int _Num> struct std::_Placeholder< _Num >
```

The type of placeholder objects defined by `libstdc++`.

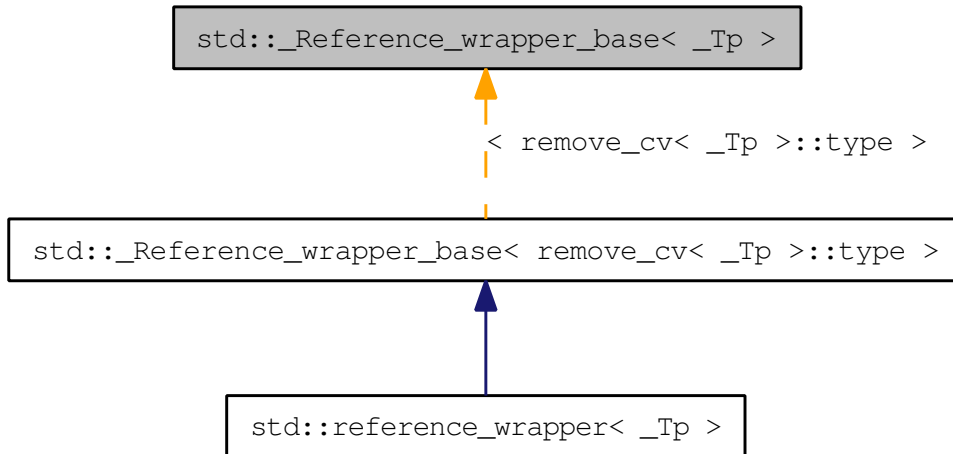
Definition at line 792 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.315 `std::_Reference_wrapper_base<_Tp>` Struct Template Reference

Inheritance diagram for `std::_Reference_wrapper_base<_Tp>`:



### 5.315.1 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](#).

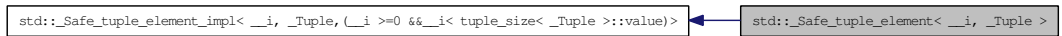
Definition at line 307 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.316 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element< __i, _Tuple >`:



### 5.316.1 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 902 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)



## 5.318 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference

### Public Types

- `typedef _No_tuple_element type`

### 5.318.1 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 892 of file `functional`.

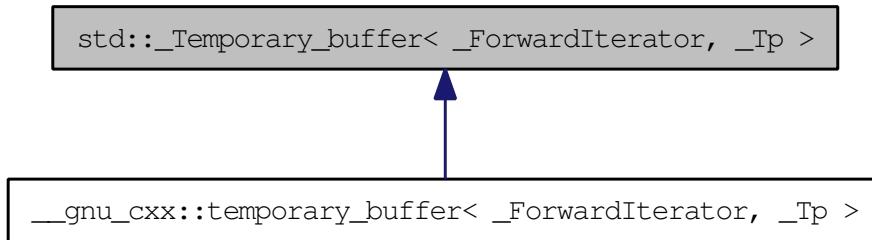
The documentation for this struct was generated from the following file:

- [functional](#)



## 5.319 `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`

### 5.319.1 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 122 of file `stl_tempbuf.h`.

### 5.319.2 Constructor & Destructor Documentation

```
5.319.2.1 template<typename _ForwardIterator , typename _Tp
> std::_Temporary_buffer< _ForwardIterator, _Tp
>::_Temporary_buffer (_ForwardIterator __first, _ForwardIterator
__last) [inline]
```

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 181 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`.

### 5.319.3 Member Function Documentation

```
5.319.3.1 template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()
[inline]
```

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

```
5.319.3.2 template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()
[inline]
```

As per Table mumble.

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

**5.319 std::\_Temporary\_buffer<\_ForwardIterator, \_Tp > Class Template Reference 1579**

---

**5.319.3.3** `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 146 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

**5.319.3.4** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp >::size() const  
[inline]`

As per Table mumble.

Definition at line 141 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](#)

## 5.320 `std::_Tuple_impl<_Idx>` Struct Template Reference

### Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &)`

### 5.320.1 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 125 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.321 `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

- `template<typename... _UElements>  
_Tuple_impl (_Tuple_impl<_Idx, _UElements...> &&__in)`
- `template<typename... _UElements>  
_Tuple_impl (const _Tuple_impl<_Idx, _UElements...> &__in)`
- `_Tuple_impl (_Tuple_impl &&__in)`
- `_Tuple_impl (const _Tuple_impl &__in)`
- `template<typename _UHead, typename... _UTail>  
_Tuple_impl (_UHead &&__head, _UTail &&...__tail)`
- `_Tuple_impl (const _Head &__head, const _Tail &...__tail)`
- `const _Head & M_head () const`
- `_Head & M_head ()`
- `const Inherited & M_tail () const`
- `Inherited & M_tail ()`
- `template<typename... _UElements>  
_Tuple_impl & operator= (_Tuple_impl<_Idx, _UElements...> &&__in)`
- `template<typename... _UElements>  
_Tuple_impl & operator= (const _Tuple_impl<_Idx, _UElements...> &__in)`
- `_Tuple_impl & operator= (_Tuple_impl &&__in)`
- `_Tuple_impl & operator= (const _Tuple_impl &__in)`

### Protected Member Functions

- `void M_swap_impl (_Tuple_impl &__in)`

### 5.321.1 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`).

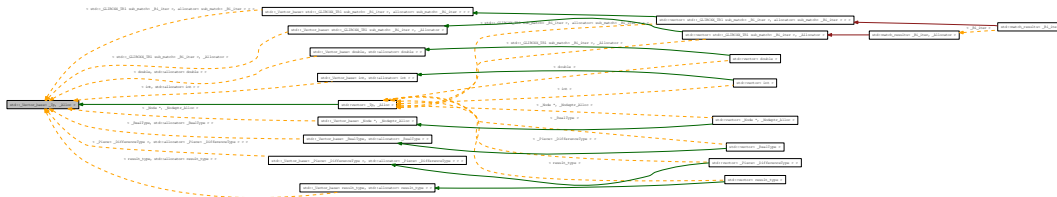
Definition at line 137 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.322 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** (**\_Vector\_base** &&\_\_x)
- **\_Vector\_base** (size\_t \_\_n, const allocator\_type &\_\_a)
- **\_Vector\_base** (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)
- const **\_Tp\_alloc\_type** & **\_M\_get\_Tp\_allocator** () const
- **\_Tp\_alloc\_type** & **\_M\_get\_Tp\_allocator** ()
- allocator\_type **get\_allocator** () const

### Public Attributes

- **\_Vector\_impl** **\_M\_impl**

#### 5.322.1 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 69 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)



## 5.323 `std::_Weak_result_type<_Functor>` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type<_Functor>`:



### 5.323.1 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 168 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.324 `std::_Weak_result_type_impl< _Functor >` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type_impl< _Functor >`:



### 5.324.1 Detailed Description

```
template<typename _Functor> struct std::_Weak_result_type_impl< _Functor
>
```

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 110 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.325 `std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function reference.

### Public Types

- `typedef _Res result_type`

#### 5.325.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>
```

Retrieve the result type for a function reference.

Definition at line 123 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.326 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function pointer.

### Public Types

- `typedef _Res result_type`

#### 5.326.1 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 130 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.327** `std::_Weak_result_type_impl<_Res(_ArgTypes...)>` Struct Template Reference 1589

---

## **5.327** `std::_Weak_result_type_impl<_Res(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function type.

### **Public Types**

- `typedef _Res result_type`

### **5.327.1 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 116 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.328 `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`

#### 5.328.1 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 144 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.329** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile > Struct` [Template Reference](#) 1591

---

## **5.329** `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`

#### **5.329.1 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 158 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.330 `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`

#### 5.330.1 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 151 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)



**5.331** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...)>` Struct  
Template Reference 1593

---

## **5.331** `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`

#### **5.331.1 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 137 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.332 `std::add_const< _Tp >` Struct Template Reference

[add\\_const](#)

### Public Types

- `typedef _Tp const type`

#### 5.332.1 Detailed Description

`template<typename _Tp> struct std::add_const< _Tp >`

[add\\_const](#)

Definition at line 426 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.333 `std::add_cv< _Tp >` Struct Template Reference

[add\\_cv](#)

### Public Types

- typedef [add\\_const](#)< typename [add\\_volatile](#)< \_Tp >::type >::type **type**

### 5.333.1 Detailed Description

`template<typename _Tp> struct std::add_cv< _Tp >`

[add\\_cv](#)

Definition at line 436 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.334 `std::add_lvalue_reference< _Tp >` Struct Template Reference

[add\\_lvalue\\_reference](#)

Inherits `std::__add_lvalue_reference_helper< _Tp >`.

### Public Types

- `typedef _Tp type`

#### 5.334.1 Detailed Description

`template<typename _Tp> struct std::add_lvalue_reference< _Tp >`

[add\\_lvalue\\_reference](#)

Definition at line 125 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.335 `std::add_pointer< _Tp >` Struct Template Reference

[add\\_pointer](#)

### Public Types

- typedef [remove\\_reference< \\_Tp >::type](#) \* **type**

### 5.335.1 Detailed Description

```
template<typename _Tp> struct std::add_pointer< _Tp >
```

[add\\_pointer](#)

Definition at line 491 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.336 `std::add_rvalue_reference< _Tp >` Struct Template Reference

[add\\_rvalue\\_reference](#)

Inherits `std::__add_rvalue_reference_helper< _Tp >`.

### Public Types

- `typedef _Tp type`

#### 5.336.1 Detailed Description

```
template<typename _Tp> struct std::add_rvalue_reference< _Tp >
```

[add\\_rvalue\\_reference](#)

Definition at line 140 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.337 `std::add_volatile< _Tp >` Struct Template Reference

[add\\_volatile](#)

### Public Types

- `typedef _Tp volatile type`

### 5.337.1 Detailed Description

```
template<typename _Tp> struct std::add_volatile< _Tp >
```

[add\\_volatile](#)

Definition at line 431 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.338 `std::adopt_lock_t` Struct Reference

Assume the calling [thread](#) has already obtained [mutex](#) ownership and manage it.

### 5.338.1 Detailed Description

Assume the calling [thread](#) has already obtained [mutex](#) ownership and manage it.

Definition at line 383 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)



## 5.339 `std::aligned_storage<_Len, _Align >` Struct Template Reference

Alignment type.

### 5.339.1 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`.

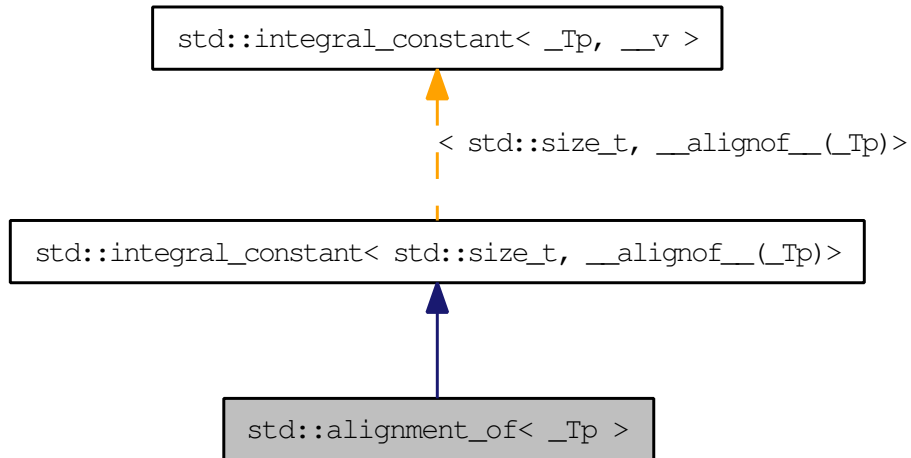
Definition at line 342 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.340 `std::alignment_of< _Tp >` Struct Template Reference

[alignment\\_of](#) Inheritance diagram for `std::alignment_of< _Tp >`:



### Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value\_type**

### Static Public Attributes

- static const `std::size_t` **value**

#### 5.340.1 Detailed Description

`template<typename _Tp> struct std::alignment_of< _Tp >`

[alignment\\_of](#)

Definition at line 350 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

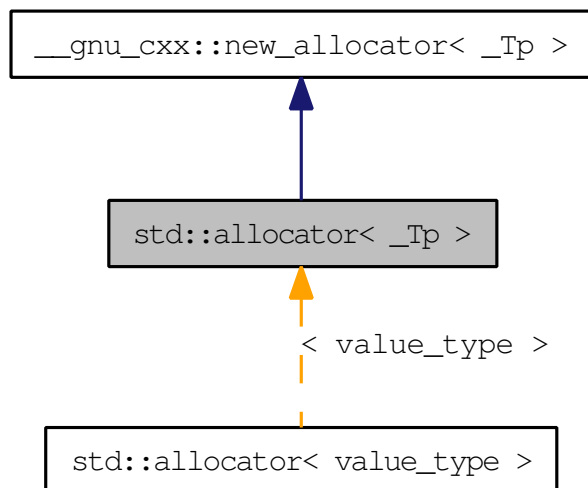
- [tr1\\_impl/type\\_traits](#)

## 5.341 `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

- `template<typename _Tp1 >`  
**allocator** (`const allocator<_Tp1 > &`) `throw ()`
- **allocator** (`const allocator &__a`) `throw ()`
- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **address** (`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)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.341.1 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 86 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.342 `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**

### 5.342.1 Detailed Description

`template<> class std::allocator< void >`

[allocator<void>](#) specialization.

Definition at line 64 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.343 `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

- const\_reference **at** (size\_type \_\_n) const
- reference **at** (size\_type \_\_n)
- const\_reference **back** () const
- reference **back** ()
- const\_iterator **begin** () const
- iterator **begin** ()
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- [const\\_reverse\\_iterator](#) **crbegin** () const
- [const\\_reverse\\_iterator](#) **crend** () const
- const \_Tp \* **data** () const
- \_Tp \* **data** ()
- bool **empty** () const
- const\_iterator **end** () const
- iterator **end** ()
- void **fill** (const value\_type &\_\_u)
- const\_reference **front** () const
- reference **front** ()
- size\_type **max\_size** () const
- const\_reference **operator[]** (size\_type \_\_n) const

- reference `operator[]` (size\_type \_\_n)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- size\_type `size` () const
- void `swap` (array &\_\_other)

## Public Attributes

- value\_type `_M_instance` [\_Nm?\_Nm:1]

### 5.343.1 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 49 of file `tr1_impl/array`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/array](#)

## 5.344 `std::atomic< _Tp >` Struct Template Reference

[atomic](#) 29.4.3, Generic [atomic](#) type, primary class template.

### Public Member Functions

- `atomic` (`_Tp __i`)
- `atomic` (`const atomic &`)
- `bool compare_exchange_strong` (`_Tp &`, `_Tp`, `memory_order=memory_order_seq_cst`) `volatile`
- `bool compare_exchange_strong` (`_Tp &`, `_Tp`, `memory_order`, `memory_order`) `volatile`
- `bool compare_exchange_weak` (`_Tp &`, `_Tp`, `memory_order=memory_order_seq_cst`) `volatile`
- `bool compare_exchange_weak` (`_Tp &`, `_Tp`, `memory_order`, `memory_order`) `volatile`
- `_Tp exchange` (`_Tp __i`, `memory_order=memory_order_seq_cst`) `volatile`
- `bool is_lock_free` () `const volatile`
- `_Tp load` (`memory_order=memory_order_seq_cst`) `const volatile`
- `operator _Tp` () `const`
- `_Tp operator=` (`_Tp __i`)
- `atomic & operator=` (`const atomic &`) `volatile`
- `void store` (`_Tp`, `memory_order=memory_order_seq_cst`) `volatile`

### 5.344.1 Detailed Description

`template<typename _Tp> struct std::atomic< _Tp >`

[atomic](#) 29.4.3, Generic [atomic](#) type, primary class template.

Definition at line 86 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)



## 5.345 `std::atomic<_Tp*>` Struct Template Reference

Partial specialization for pointer types.

Inherits `atomic_address`.

### Public Member Functions

- `atomic` (`_Tp *__v`)
- `atomic` (const `atomic` &)
- bool `compare_exchange_strong` (`_Tp * &`, `_Tp *`, `memory_order=memory_order_seq_cst`)
- bool `compare_exchange_strong` (`_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`)
- `_Tp *` `exchange` (`_Tp *`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `fetch_add` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `fetch_sub` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `load` (`memory_order=memory_order_seq_cst`) const
- `operator _Tp *` () const
- `_Tp *` `operator++` ()
- `_Tp *` `operator++` (int)
- `_Tp *` `operator+=` (`ptrdiff_t __d`)
- `_Tp *` `operator--` ()
- `_Tp *` `operator--` (int)
- `_Tp *` `operator-=` (`ptrdiff_t __d`)
- `_Tp *` `operator=` (`_Tp *__v`)
- `atomic` & `operator=` (const `atomic` &) volatile
- void `store` (`_Tp *`, `memory_order=memory_order_seq_cst`)

### 5.345.1 Detailed Description

`template<typename _Tp> struct std::atomic<_Tp*>`

Partial specialization for pointer types.

Definition at line 134 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.346 `std::atomic< bool >` Struct Template Reference

Explicit specialization for `bool`.

Inherits `__atomic_bool_base`.

### Public Types

- `typedef atomic_bool __base_type`
- `typedef bool __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `bool _M_i`

#### 5.346.1 Detailed Description

`template<> struct std::atomic< bool >`

Explicit specialization for `bool`.

Definition at line 222 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.347 `std::atomic< char >` Struct Template Reference

Explicit specialization for `char`.

Inherits `__atomic_char_base`.

### Public Types

- typedef `atomic_char_base_type`
- typedef `char __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `char _M_i`

#### 5.347.1 Detailed Description

`template<> struct std::atomic< char >`

Explicit specialization for `char`.

Definition at line 240 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.348 `std::atomic< char16_t >` Struct Template Reference

Explicit specialization for `char16_t`.

Inherits `__atomic_short_base`.

### Public Types

- typedef `atomic_char16_t_base_type`
- typedef `char16_t __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

### Public Attributes

- `short _M_i`

#### 5.348.1 Detailed Description

`template<> struct std::atomic< char16_t >`

Explicit specialization for `char16_t`.

Definition at line 456 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.349 `std::atomic< char32_t >` Struct Template Reference

Explicit specialization for `char32_t`.

Inherits `__atomic_int_base`.

### Public Types

- typedef `atomic_char32_t_base_type`
- typedef `char32_t_integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `int_M_i`

#### 5.349.1 Detailed Description

`template<> struct std::atomic< char32_t >`

Explicit specialization for `char32_t`.

Definition at line 474 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.350 `std::atomic< int >` Struct Template Reference

Explicit specialization for `int`.

Inherits `__atomic_int_base`.

### Public Types

- typedef `atomic_int_base_type`
- typedef `int __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `int _M_i`

#### 5.350.1 Detailed Description

`template<> struct std::atomic< int >`

Explicit specialization for `int`.

Definition at line 330 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.351 `std::atomic< long >` Struct Template Reference

Explicit specialization for long.

Inherits `__atomic_long_base`.

### Public Types

- typedef `atomic_long_base_type`
- typedef long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- `long _M_i`

#### 5.351.1 Detailed Description

`template<> struct std::atomic< long >`

Explicit specialization for long.

Definition at line 366 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.352 `std::atomic< long long >` Struct Template Reference

Explicit specialization for long long.

Inherits `__atomic_llong_base`.

### Public Types

- typedef `atomic_llong __base_type`
- typedef long long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

### Public Attributes

- long long `_M_i`

#### 5.352.1 Detailed Description

`template<> struct std::atomic< long long >`

Explicit specialization for long long.

Definition at line 402 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`



## 5.353 `std::atomic< short >` Struct Template Reference

Explicit specialization for short.

Inherits `__atomic_short_base`.

### Public Types

- typedef `atomic_short_base_type`
- typedef short `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- short `_M_i`

#### 5.353.1 Detailed Description

`template<> struct std::atomic< short >`

Explicit specialization for short.

Definition at line 294 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.354 `std::atomic< signed char >` Struct Template Reference

Explicit specialization for signed char.

Inherits `__atomic_schar_base`.

### Public Types

- typedef `atomic_schar_base_type`
- typedef signed char `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- signed char `_M_i`

#### 5.354.1 Detailed Description

`template<> struct std::atomic< signed char >`

Explicit specialization for signed char.

Definition at line 258 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.355 `std::atomic< unsigned char >` Struct Template Reference

Explicit specialization for unsigned char.

Inherits `__atomic_uchar_base`.

### Public Types

- typedef `atomic_uchar __base_type`
- typedef `unsigned char __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `unsigned char _M_i`

#### 5.355.1 Detailed Description

`template<> struct std::atomic< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 276 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.356 `std::atomic< unsigned int >` Struct Template Reference

Explicit specialization for unsigned int.

Inherits `__atomic_uint_base`.

### Public Types

- typedef `atomic_uint __base_type`
- typedef unsigned int `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- unsigned int `_M_i`

#### 5.356.1 Detailed Description

`template<> struct std::atomic< unsigned int >`

Explicit specialization for unsigned int.

Definition at line 348 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.357 `std::atomic< unsigned long >` Struct Template Reference

Explicit specialization for unsigned long.

Inherits `__atomic_ulong_base`.

### Public Types

- typedef `atomic_ulong` `__base_type`
- typedef unsigned long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- unsigned long `_M_i`

#### 5.357.1 Detailed Description

`template<> struct std::atomic< unsigned long >`

Explicit specialization for unsigned long.

Definition at line 384 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.358 `std::atomic< unsigned long long >` Struct Template Reference

Explicit specialization for unsigned long long.

Inherits `__atomic_ullong_base`.

### Public Types

- typedef `atomic_ullong_base_type`
- typedef unsigned long long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- unsigned long long `_M_i`

#### 5.358.1 Detailed Description

`template<> struct std::atomic< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 420 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.359 `std::atomic< unsigned short >` Struct Template Reference

Explicit specialization for unsigned short.

Inherits `__atomic_ushort_base`.

### Public Types

- typedef `atomic_ushort_base_type`
- typedef unsigned short `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- unsigned short `_M_i`

### 5.359.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 312 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.360 `std::atomic< void * >` Struct Template Reference

Explicit specialization for `void*`.

Inherits `atomic_address`.

### Public Types

- `typedef atomic_address __base_type`
- `typedef void * __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

#### 5.360.1 Detailed Description

`template<> struct std::atomic< void * >`

Explicit specialization for `void*`.

Definition at line 204 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`



## 5.361 `std::atomic< wchar_t >` Struct Template Reference

Explicit specialization for `wchar_t`.

Inherits `__atomic_wchar_t_base`.

### Public Types

- typedef `atomic_wchar_t_base_type`
- typedef `wchar_t_integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

### Public Attributes

- `wchar_t_M_i`

#### 5.361.1 Detailed Description

`template<> struct std::atomic< wchar_t >`

Explicit specialization for `wchar_t`.

Definition at line 438 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.362 `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` (`auto_ptr_ref< element_type > __ref`) throw ()
- `template<typename _Tp1 >`  
`auto_ptr` (`auto_ptr<_Tp1 > &__a`) throw ()
- `auto_ptr` (`auto_ptr &__a`) throw ()
- `auto_ptr` (`element_type *__p=0`) 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_ref< element_type > __ref`) throw ()
- `template<typename _Tp1 >`  
`auto_ptr & operator=` (`auto_ptr<_Tp1 > &__a`) throw ()
- `auto_ptr & operator=` (`auto_ptr &__a`) throw ()
- `element_type * release` () throw ()
- `void reset` (`element_type *__p=0`) throw ()

### 5.362.1 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 CopyCon 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++` test suite.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 85 of file `auto_ptr.h`.

## 5.362.2 Member Typedef Documentation

### 5.362.2.1 `template<typename _Tp> typedef _Tp std::auto_ptr<_Tp>::element_type`

The pointed-to type.

Definition at line 92 of file `auto_ptr.h`.

## 5.362.3 Constructor & Destructor Documentation

### 5.362.3.1 `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 101 of file `auto_ptr.h`.

### 5.362.3.2 `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 110 of file `auto_ptr.h`.

**5.362.3.3** `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 123 of file `auto_ptr.h`.

**5.362.3.4** `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 168 of file `auto_ptr.h`.

**5.362.3.5** `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 258 of file `auto_ptr.h`.

## 5.362.4 Member Function Documentation

### 5.362.4.1 `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 209 of file auto\_ptr.h.

### 5.362.4.2 `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 179 of file auto\_ptr.h.

### 5.362.4.3 `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 192 of file auto\_ptr.h.

### 5.362.4.4 `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 152 of file auto\_ptr.h.

```
5.362.4.5 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 134 of file auto\_ptr.h.

```
5.362.4.6 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 223 of file auto\_ptr.h.

```
5.362.4.7 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 238 of file auto\_ptr.h.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

## 5.363 `std::auto_ptr_ref< _Tp1 >` Struct Template Reference

### Public Member Functions

- `auto_ptr_ref(_Tp1 * __p)`

### Public Attributes

- `_Tp1 * _M_ptr`

#### 5.363.1 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](#) by value.

All the [auto\\_ptr\\_ref](#) stuff should happen behind the scenes.

Definition at line 46 of file `auto_ptr.h`.

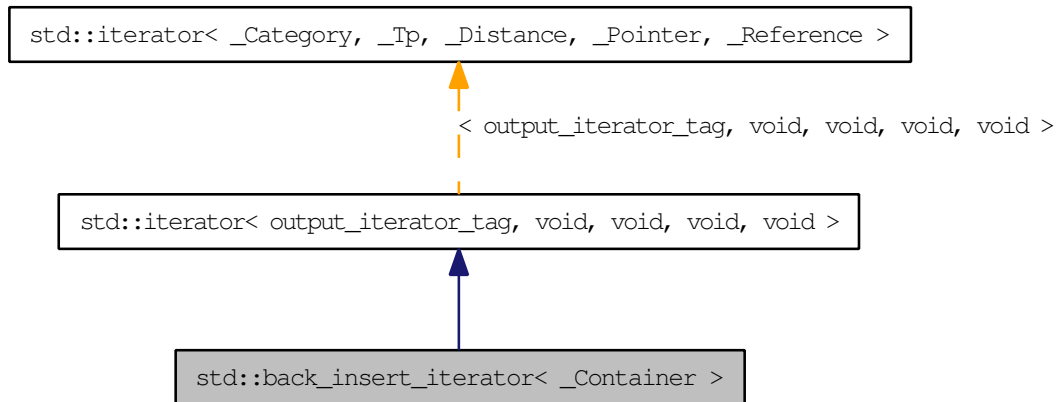
The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)



## 5.364 `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++` (int)
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_-value)
- `back_insert_iterator` & `operator=` (typename `_Container::const_reference` \_\_-value)

## Protected Attributes

- `_Container * container`

### 5.364.1 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 394 of file `stl_iterator.h`.

### 5.364.2 Member Typedef Documentation

**5.364.2.1** `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 402 of file `stl_iterator.h`.

**5.364.2.2** `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 115 of file `stl_iterator_base_types.h`.

**5.364.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void  
, void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 111 of file `stl_iterator_base_types.h`.

**5.364.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 117 of file `stl_iterator_base_types.h`.

## **5.364 std::back\_insert\_iterator< \_Container > Class Template Reference 1635**

**5.364.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file stl\_iterator\_base\_types.h.

**5.364.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file stl\_iterator\_base\_types.h.

### **5.364.3 Constructor & Destructor Documentation**

**5.364.3.1** `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 406 of file stl\_iterator.h.

### **5.364.4 Member Function Documentation**

**5.364.4.1** `template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator*() [inline]`

Simply returns \*this.

Definition at line 437 of file stl\_iterator.h.

**5.364.4.2** `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 447 of file stl\_iterator.h.

**5.364.4.3** `template<typename _Container > back_insert_iterator&  
std::back_insert_iterator< _Container >::operator++ () [inline]`

Simply returns \*this. (This iterator does not *move*.)

Definition at line 442 of file stl\_iterator.h.

**5.364.4.4** `template<typename _Container > back_insert_iterator&  
std::back_insert_iterator< _Container >::operator= (typename  
_Container::const_reference __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 420 of file stl\_iterator.h.

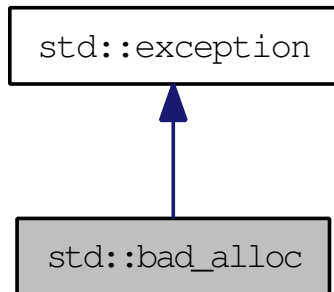
The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.365 `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 ()

#### 5.365.1 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`.

#### 5.365.2 Member Function Documentation

##### 5.365.2.1 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`.

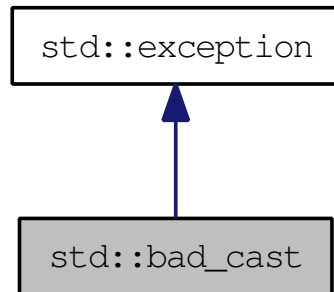
The documentation for this class was generated from the following file:

- `new`

## 5.366 `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 ()

#### 5.366.1 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 174 of file `typeinfo`.

#### 5.366.2 Member Function Documentation

##### 5.366.2.1 virtual const char\* `std::bad_cast::what` () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

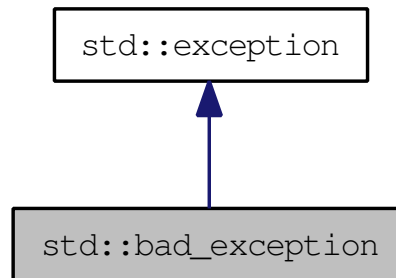
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.367 std::bad\_exception Class Reference

Inheritance diagram for std::bad\_exception:



### Public Member Functions

- virtual const char \* [what](#) () const throw ()

#### 5.367.1 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.

#### 5.367.2 Member Function Documentation

##### 5.367.2.1 virtual const char\* std::bad\_exception::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

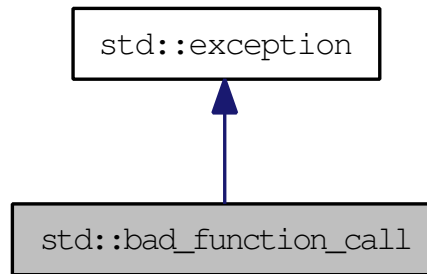
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

## 5.368 `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 ()

#### 5.368.1 Detailed Description

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

Definition at line 1405 of file `functional`.

#### 5.368.2 Member Function Documentation

##### 5.368.2.1 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::bad\\_weak\\_ptr](#), and [std::ios\\_base::failure](#).

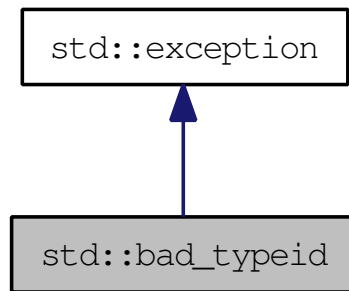
The documentation for this class was generated from the following file:

- [functional](#)



## 5.369 `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 ()

#### 5.369.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 191 of file `typeinfo`.

#### 5.369.2 Member Function Documentation

##### 5.369.2.1 virtual const char\* `std::bad_typeid::what` () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

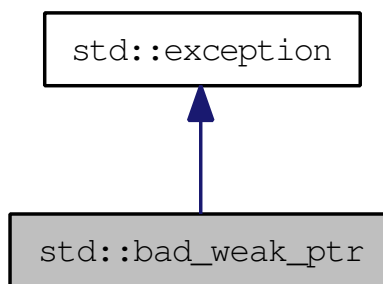
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.370 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 ()

#### 5.370.1 Detailed Description

Exception possibly thrown by [shared\\_ptr](#).

Definition at line 58 of file boost\_sp\_counted\_base.h.

#### 5.370.2 Member Function Documentation

##### 5.370.2.1 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 62 of file boost\_sp\_counted\_base.h.

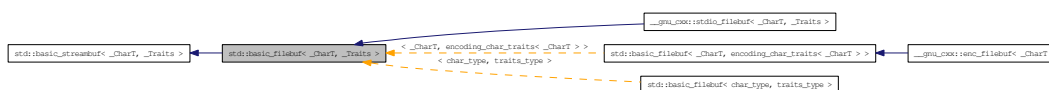
The documentation for this class was generated from the following file:

- [boost\\_sp\\_counted\\_base.h](#)

## 5.371 `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 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)`
- `void stoss ()`
- `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 ()`
- `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)`

## 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, 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` `_M_buf_locale`
- `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`
- `char_type` \* `eback` () const
- `char_type` \* `egptr` () const
- `char_type` \* `epptr` () const
- `char_type` \* `gptr` () const
- `char_type` \* `pbase` () const
- `char_type` \* `pptr` () const
- `locale` `getloc` () const
- `locale` `pubimbue` (const `locale` &\_\_loc)
- `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`)
- `__streambuf_type` \* `pubsetbuf` (`char_type` \*\_\_s, `streamsize` \_\_n)
- `int` `pubsync` ()

### 5.371.1 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 67 of file `fstream`.

### 5.371.2 Member Typedef Documentation

```
5.371.2.1 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 77 of file `fstream`.

**5.371.2.2** `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 71 of file `fstream`.

**5.371.2.3** `template<typename _CharT, typename _Traits> typedef  
traits_type::int_type std::basic_filebuf<_CharT, _Traits>::int_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 73 of file `fstream`.

**5.371.2.4** `template<typename _CharT, typename _Traits> typedef  
traits_type::off_type std::basic_filebuf<_CharT, _Traits>::off_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 75 of file `fstream`.

**5.371.2.5** `template<typename _CharT, typename _Traits> typedef  
traits_type::pos_type std::basic_filebuf<_CharT, _Traits  
>::pos_type`

This is a non-standard type.

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.

**5.371.2.6** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_filebuf< _CharT, _Traits >::traits_type`

This is a non-standard type.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Reimplemented in `__gnu_cxx::enc_filebuf< _CharT >`, and `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`.

Definition at line 72 of file fstream.

### 5.371.3 Constructor & Destructor Documentation

**5.371.3.1** `template<typename _CharT, typename _Traits > std::basic_filebuf<  
_CharT, _Traits >::basic_filebuf () [inline]`

Does not open any files. The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file fstream.tcc.

References `std::basic_streambuf< _CharT, _Traits >::_M_buf_locale`.

**5.371.3.2** `template<typename _CharT, typename _Traits> virtual  
std::basic_filebuf< _CharT, _Traits >::~basic_filebuf () [inline,  
virtual]`

The destructor closes the file first.

Definition at line 214 of file fstream.

### 5.371.4 Member Function Documentation

**5.371.4.1** `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 172 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`.



**5.371.4.2** `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 189 of file fstream.

Referenced by 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().

**5.371.4.3** `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: `egptr() - 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 387 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), `_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().

**5.371.4.4** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,  
_Traits >::close () [inline]`

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 128 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

#### 5.371.4.5 `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 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

#### 5.371.4.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr() const [inline, protected, 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 466 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.371.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eptr () const  
[inline, protected, 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 513 of file `streambuf`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.371.4.8** `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 476 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.371.4.9** `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 222 of file streambuf.

```
5.371.4.10 template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected, 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 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

```
5.371.4.11 template<typename _CharT , typename _Traits > void
std::basic_filebuf< _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 from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 855 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().

#### 5.371.4.12 `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 262 of file streambuf.

#### 5.371.4.13 `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 222 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::close(), std::basic\_filebuf< \_CharT, \_Traits >::imbue(), 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()`.

#### 5.371.4.14 `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 275 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

**5.371.4.15** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (const char * __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

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)

| <code>ios_base</code> Flag combination | stdio equivalent | binary | in | out | trunc | app |
|----------------------------------------|------------------|--------|----|-----|-------|-----|
| <code>r</code>                         | <code>r</code>   |        |    |     |       |     |
| <code>w</code>                         | <code>w</code>   |        |    |     |       |     |
| <code>a</code>                         | <code>a</code>   |        |    |     |       |     |
| <code>w+</code>                        | <code>w+</code>  |        |    |     |       |     |
| <code>a+</code>                        | <code>a+</code>  |        |    |     |       |     |
| <code>+</code>                         | <code>+</code>   |        |    |     |       |     |
| <code>rb</code>                        | <code>rb</code>  |        |    |     |       |     |
| <code>rb+</code>                       | <code>rb+</code> |        |    |     |       |     |
| <code>wb</code>                        | <code>wb</code>  |        |    |     |       |     |
| <code>wb+</code>                       | <code>wb+</code> |        |    |     |       |     |
| <code>ab</code>                        | <code>ab</code>  |        |    |     |       |     |
| <code>ab+</code>                       | <code>ab+</code> |        |    |     |       |     |
| <code>+</code>                         | <code>+</code>   |        |    |     |       |     |

Definition at line 94 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()`.

**5.371.4.16** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow(int_type = _Traits::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 408 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::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 >::sync().

**5.371.4.17** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail(int_type = _Traits::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 356 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::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>::seekoff\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#).

#### 5.371.4.18 **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 507 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\\_filebuf<\\_CharT, \\_Traits>::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).



**5.371.4.19** `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 523 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()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.371.4.20** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr () const  
[inline, protected, 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 510 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_filebuf< _CharT, _Traits >::sync()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.371.4.21** `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 205 of file `streambuf`.

**5.371.4.22** `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]`

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 239 of file `streambuf`.

**5.371.4.23** `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]`

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 244 of file `streambuf`.

**5.371.4.24** `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 235 of file streambuf.

**5.371.4.25** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync () [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 249 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.371.4.26** `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 294 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

**5.371.4.27** `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) [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 from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 686 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_filebuf<_CharT, _Traits>::pbackfail()`.

**5.371.4.28** `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) [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 from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 739 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()`.

**5.371.4.29** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::__streambuf_type * std::basic_filebuf<_CharT, _Traits>::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 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 657 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()`.

**5.371.4.30** `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 487 of file `streambuf`.

**5.371.4.31** `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 533 of file streambuf.

**5.371.4.32** `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 316 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()`.

**5.371.4.33** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s,  
streamsize __n) [inline, inherited]`

Entry point for `xsggetn`.

**Parameters:**

*s* A buffer area.

*n* A count.

Returns `xsggetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

**5.371.4.34** `template<typename _CharT , typename _Traits > streamsize  
std::basic_filebuf< _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.*

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 178 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()`.

#### 5.371.4.35 `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 276 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()`.

**5.371.4.36** `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 350 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.371.4.37** `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 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.



**5.371.4.38** `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 `xsputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.371.4.39** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stoss() [inline,  
inherited]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.371.4.40** `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 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.371.4.41** `template<typename _CharT, typename _Traits> int  
std::basic_filebuf<_CharT, _Traits>::sync() [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 838 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

**5.371.4.42** `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 678 of file `streambuf`.

**5.371.4.43** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits>::int_type std::basic_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 204 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`, and `std::min()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`.

#### 5.371.4.44 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn(char_type * __s, streamsize __n) [inline, 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 527 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`, and `std::basic_streambuf<char_type, traits_type>::xsgetn()`.

**5.371.4.45** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::xsputn(const char_type *  
__s, streamsize __n) [inline, 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 610 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`, and `std::basic_streambuf<_`

`CharT, _Traits >::pbase()`, `std::basic_streambuf<_CharT, _Traits >::pptr()`, and `std::basic_streambuf<char_type, traits_type >::xsputn()`.

### 5.371.5 Member Data Documentation

#### 5.371.5.1 `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 109 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::setbuf()`.

#### 5.371.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits >::_M_buf_locale` [protected, inherited]

Current `locale` setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::basic_filebuf()`.

#### 5.371.5.3 `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 116 of file `fstream`.

Referenced by `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()`.

#### 5.371.5.4 `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 151 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `std::basic_filebuf<_CharT, _Traits >::seekoff()`, and `std::basic_filebuf<_CharT, _Traits >::underflow()`.

**5.371.5.5** `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 156 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.371.5.6** `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 163 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.371.5.7** `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 180 of file `streambuf`.

**5.371.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, 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 181 of file streambuf.

**5.371.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, 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 182 of file streambuf.

**5.371.5.10** `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 94 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 >::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()`.

**5.371.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, 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 183 of file streambuf.

**5.371.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, 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 184 of file `streambuf`.

**5.371.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, 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 185 of file `streambuf`.

**5.371.5.14** `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 137 of file `fstream`.

**5.371.5.15** `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 138 of file fstream.

**5.371.5.16** `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 139 of file fstream.

**5.371.5.17** `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 140 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.371.5.18** `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 128 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()`.

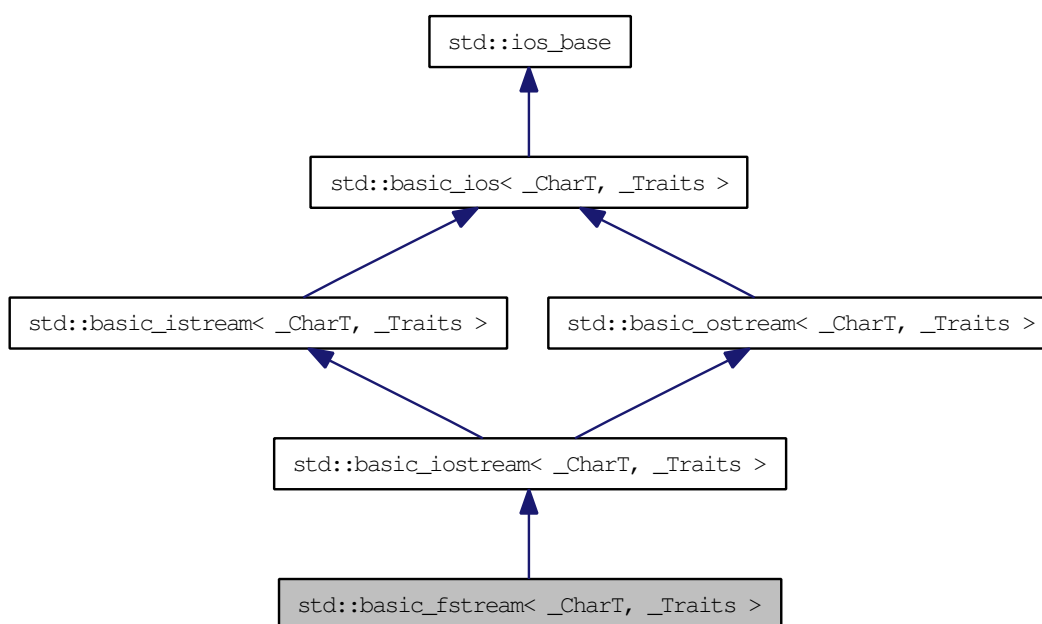
The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)

## 5.372 `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_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_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< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_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**
- 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**

## Public Member Functions

- [basic\\_fstream](#) (const [std::string](#) &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [basic\\_fstream](#) (const char \*\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [basic\\_fstream](#) ()
- [~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
- void [exceptions](#) (iostate \_\_except)
- iostate [exceptions](#) () const
- bool [fail](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [char\\_type](#) [fill](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const

- `__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, int_type __delim)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `locale imbue (const locale &__loc)`
- `bool is_open () const`
- `bool is_open ()`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize precision (streamsize __prec)`
- `streamsize precision () const`
- `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 (off_type, ios_base::seekdir)`
- `__ostream_type & seekp (pos_type)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `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

### 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.

- void [\\_M\\_write](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [put](#) ([char\\_type](#) \_\_c)
- [\\_\\_ostream\\_type](#) & [write](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)

### 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.

- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) &\_\_c)
- [int\\_type](#) [get](#) ()
- [\\_\\_istream\\_type](#) & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)

- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__istream_type & seekg (pos_type)`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & unget ()`

### 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<< (__streambuf_type *__sb)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`
  
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`

### 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>> ([\\_\\_streambuf\\_type](#) \* \_\_sb)
- [\\_\\_istream\\_type](#) & operator>> (void \*&\_\_p)
- [\\_\\_istream\\_type](#) & operator>> (long double &\_\_f)
- [\\_\\_istream\\_type](#) & operator>> (double &\_\_f)
- [\\_\\_istream\\_type](#) & operator>> (float &\_\_f)
- [\\_\\_istream\\_type](#) & operator>> (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (long long &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (long &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (int &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (short &\_\_n)
- [\\_\\_istream\\_type](#) & operator>> (bool &\_\_n)
  
- [\\_\\_istream\\_type](#) & operator>> ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))
- [\\_\\_istream\\_type](#) & operator>> ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_istream\\_type](#) & operator>> ([\\_\\_istream\\_type](#) &(\*\_\_pf)([\\_\\_istream\\_type](#) &))

## 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 `iosate eofbit`
- static const `iosate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosate 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`
  
- `typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- `operator void * () const`
- `bool operator! () const`

### 5.372.1 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 `s.b.`

Definition at line 756 of file `fstream`.

## 5.372.2 Member Typedef Documentation

**5.372.2.1** `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 71 of file `ostream`.

**5.372.2.2** `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 71 of file `istream`.

**5.372.2.3** `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 70 of file `istream`.

**5.372.2.4** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ostream<_CharT, _Traits>::_num_put_type [inherited]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `ostream`.

**5.372.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::_num_put_type` **[inherited]**

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

**5.372.2.6** `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 760 of file `fstream`.

**5.372.2.7** `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 444 of file `ios_base.h`.

**5.372.2.8** `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 263 of file `ios_base.h`.

**5.372.2.9** `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_fstream< _CharT, _Traits >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 762 of file `fstream`.

**5.372.2.10 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 338 of file `ios_base.h`.

**5.372.2.11 template<typename `_CharT`, typename `_Traits`> typedef `traits_type::off_type` `std::basic_fstream<_CharT, _Traits>::off_type`**

These are non-standard types.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 764 of file `fstream`.

**5.372.2.12 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 369 of file `ios_base.h`.

**5.372.2.13** `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_fstream<_CharT, _Traits>::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 763 of file `fstream`.

**5.372.2.14** `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 401 of file `ios_base.h`.

**5.372.2.15** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_fstream<_CharT, _Traits>::traits_type`

These are non-standard types.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 761 of file `fstream`.

## 5.372.3 Member Enumeration Documentation

**5.372.3.1** `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 427 of file `ios_base.h`.

## 5.372.4 Constructor & Destructor Documentation

**5.372.4.1** `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 783 of file `fstream`.

**5.372.4.2** `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 796 of file `fstream`.

**5.372.4.3** `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 811 of file `fstream`.



**5.372.4.4** `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 826 of file fstream.

## 5.372.5 Member Function Documentation

**5.372.5.1** `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 705 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()`.

**5.372.5.2** `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 287 of file ostream.

Referenced by `std::basic_ostream<_CharT, _Traits>::write()`.

**5.372.5.3** `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 201 of file basic\_ios.h.

**5.372.5.4** `template<typename _CharT , typename _Traits > void std::basic_ios<_CharT, _Traits >::clear (iostate __state = goodbit)` `[inline, 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 40 of file basic\_ios.tcc.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.372.5.5** `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 906 of file fstream.

**5.372.5.6** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt(const basic_ios<_CharT, _Traits> & __rhs) [inline, 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 62 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()`.

**5.372.5.7** `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 180 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()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.372.5.8** `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 247 of file `basic_ios.h`.

#### 5.372.5.9 `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 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

#### 5.372.5.10 `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 191 of file `basic_ios.h`.

Referenced by `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()`.

#### 5.372.5.11 `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 380 of file `basic_ios.h`.

#### 5.372.5.12 `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.372.5.13** `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 561 of file `ios_base.h`.

**5.372.5.14** `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 550 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<<()`, and `std::operator>>()`.

**5.372.5.15** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inline, 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 211 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()`.

**5.372.5.16** `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 249 of file `istream`.

**5.372.5.17** `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 366 of file `istream`.

**5.372.5.18** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get(  
__streambuf_type & __sb, char_type __delim) [inline,  
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 356 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()`.

**5.372.5.19** `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 333 of file istream.



**5.372.5.20** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get(char_type * __s, streamsize __n, char_type __delim) [inline, 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 309 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()`.

**5.372.5.21** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get(char_type & __c) [inline, 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 272 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()`.

**5.372.5.22** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get(void) [inline, 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 236 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()`.

**5.372.5.23** `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 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.372.5.24** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline(char_type * __s, streamsize __n, char_type __delim) [inline, 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 400 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()`.

#### 5.372.5.25 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 694 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()`.

#### 5.372.5.26 `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 170 of file `basic_ios.h`.

#### 5.372.5.27 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim)` [`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 555 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()`.

#### 5.372.5.28 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n) [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 493 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()`.

#### 5.372.5.29 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(void) [inline, 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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.372.5.30** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file basic\_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.372.5.31** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<_CharT,  
_Traits> * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.372.5.32** `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 845 of file `fstream`.

**5.372.5.33** `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 740 of file `ios_base.h`.

**5.372.5.34** `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/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.372.5.35** `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 887 of file `fstream`.

**5.372.5.36** `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 866 of file `fstream`.

**5.372.5.37** `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 111 of file `basic_ios.h`.

**5.372.5.38** `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 115 of file `basic_ios.h`.

**5.372.5.39** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (_streambuf_type * __sb)` [`inline`, `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 120 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()`.

**5.372.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (const void * __p) [inline, 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 225 of file ostream.

**5.372.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (long double __f) [inline, 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 221 of file ostream.

**5.372.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (float __f) [inline, 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 213 of file ostream.

**5.372.5.43** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (double __f) [inline, 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 209 of file ostream.

**5.372.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long long __n) [inline, 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 204 of file `ostream`.

**5.372.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, 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 200 of file `ostream`.

**5.372.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned int __n) [inline, 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 191 of file ostream.

**5.372.5.47** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (int __n) [inline, 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 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.372.5.48** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned short __n) [inline, 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 180 of file `ostream`.

**5.372.5.49** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (short __n) [inline, 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 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.372.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline, 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 173 of file ostream.

**5.372.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long __n) [inline, 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 169 of file ostream.

**5.372.5.52** `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 165 of file ostream.

**5.372.5.53** `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 127 of file ostream.

**5.372.5.54** `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 117 of file ostream.

**5.372.5.55** `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 108 of file ostream.

**5.372.5.56** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inline, 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 204 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()`.

**5.372.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (void *& __p) [inline, 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 215 of file istream.

**5.372.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, 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 211 of file istream.

**5.372.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline, 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 207 of file `istream`.

**5.372.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (float & __f) [inline, 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 203 of file `istream`.

**5.372.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, 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 198 of file istream.

**5.372.5.62** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline, 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 194 of file `istream`.

**5.372.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline, 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 189 of file `istream`.

**5.372.5.64** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long & __n) [inline, 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 185 of file istream.

**5.372.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, 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 181 of file istream.



**5.372.5.66** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inline, 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 159 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()`.

**5.372.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, 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 174 of file istream.

**5.372.5.68** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (short & __n) [inline, 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 114 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()`.

**5.372.5.69** `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 167 of file `istream`.

**5.372.5.70** `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 131 of file `istream`.

**5.372.5.71** `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 124 of file `istream`.

**5.372.5.72** `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 120 of file `istream`.

**5.372.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek(void) [inline, 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 620 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()`.

**5.372.5.74** `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 629 of file `ios_base.h`.

**5.372.5.75** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.372.5.76** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c) [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 149 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()`.

**5.372.5.77** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (char_type __c) [inline, 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 711 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

#### 5.372.5.78 `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 761 of file `ios_base.h`.

#### 5.372.5.79 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf(basic_streambuf<_CharT, _Traits> * __sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

#### 5.372.5.80 `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 837 of file `fstream`.

#### 5.372.5.81 `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 127 of file `basic_ios.h`.

#### 5.372.5.82 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(char_type* __s, streamsize __n)` [`inline`, `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 650 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()`.

**5.372.5.83** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome(char_type * __s,  
streamsize __n) [inline, 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 679 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()`.

#### 5.372.5.84 `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.

#### 5.372.5.85 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir) [inline, 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 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.372.5.86** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::seekg(pos_type __pos) [inline, 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 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.372.5.87** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::seekp(off_type __off, ios_base::seekdir __dir) [inline, inherited]`

Changing the current write position.

**Parameters:**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns:**

\*this

## 5.372 `std::basic_fstream<_CharT, _Traits>` Class Template Reference 1731

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 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()`.

### 5.372.5.88 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos)` [`inline`, `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 258 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()`.

### 5.372.5.89 `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 594 of file `ios_base.h`.

### 5.372.5.90 `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

### 5.372.5.91 `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 147 of file `basic_ios.h`.

Referenced by `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_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 >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.372.5.92** `template<typename _CharT, typename _Traits> int  
std::basic_istream<_CharT, _Traits>::sync(void) [inline,  
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 777 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()`.

**5.372.5.93** `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>

**5.372.5.94** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inline, 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 813 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()`.

**5.372.5.95** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inline, 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 237 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()`.

**5.372.5.96** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tistr) [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 297 of file `basic_ios.h`.

**5.372.5.97** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.372.5.98** `template<typename _CharT, typename _Traits> basic_istream<`  
`_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget`  
`(void) [inline, 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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.372.5.99** `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 609 of file ios\_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.372.5.100** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic\_ios.h.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.



**5.372.5.101** `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 652 of file `ios_base.h`.

**5.372.5.102** `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 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator<>>()`.

**5.372.5.103** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inline, 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 183 of file `ostream.tcc`.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.372.5.104 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.

**5.372.6 Member Data Documentation****5.372.6.1 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 79 of file `istream`.

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 >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.372.6.2 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 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

**5.372.6.3 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.372.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.372.6.5 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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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 >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_ostream< _CharT, _Traits >::write()`.

**5.372.6.6 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 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.372.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::seekpos()`.

**5.372.6.8 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 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

**5.372.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

**5.372.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `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()`.

**5.372.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file ios\_base.h.

**5.372.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.372.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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\_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 >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

**5.372.6.14 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 350 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\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), 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(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.372.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.372.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios\_base.h.

**5.372.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::sync()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.372.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.372.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

## 5.372 `std::basic_fstream<_CharT, _Traits>` Class Template Reference 1743

Referenced by `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_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()`.

### 5.372.6.20 `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 280 of file `ios_base.h`.

### 5.372.6.21 `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter>::do_put()`.

### 5.372.6.22 `const fmtflags std::ios_base::oct` [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

### 5.372.6.23 `const openmode std::ios_base::out` [static, inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `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()`.

**5.372.6.24 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 291 of file ios\_base.h.

**5.372.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.372.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file ios\_base.h.

**5.372.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file ios\_base.h.

**5.372.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.372.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.372.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.



**5.372.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.372.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

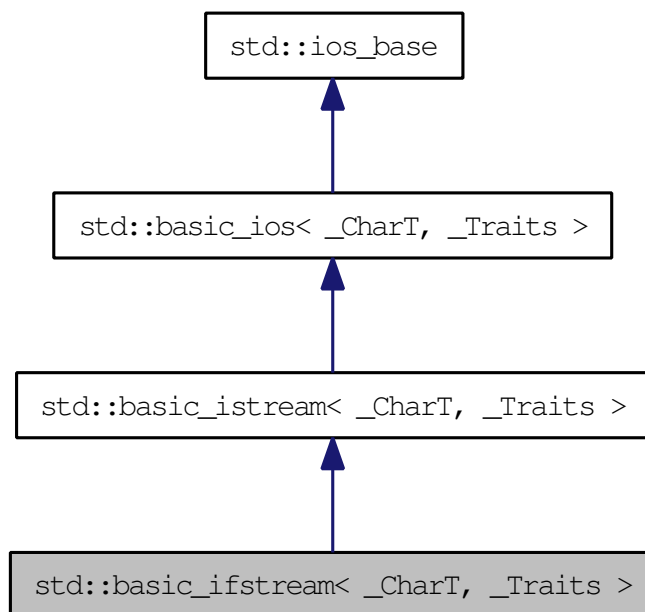
The documentation for this class was generated from the following file:

- [fstream](#)

## 5.373 `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`

## Public Member Functions

- `basic_ifstream` (`const std::string &__s`, `ios_base::openmode __mode=ios_base::in`)
- `basic_ifstream` (`const char *__s`, `ios_base::openmode __mode=ios_base::in`)
- `basic_ifstream` ()
- `~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
- `void exceptions` (`iostate __except`)
- `iostate exceptions` () const
- `bool fail` () const
- `char_type fill` (`char_type __ch`)
- `char_type fill` () const
- `fmtflags flags` (`fmtflags __fmtfl`)
- `fmtflags flags` () const
- `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, [int\\_type](#) \_\_delim)
- [template<>](#)  
[basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [template<>](#)  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [template<>](#)  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [locale imbue](#) (const [locale](#) & \_\_loc)
- [bool is\\_open](#) () const
- [bool is\\_open](#) ()
- [long & iword](#) (int \_\_ix)
- [char narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [void open](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [void open](#) (const [char](#) \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [streamsize precision](#) ([streamsize](#) \_\_prec)
- [streamsize precision](#) () const
- [void \\*& pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- [\\_\\_filebuf\\_type](#) \* [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](#) ([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

### 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.

- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type &__c)`
- `int_type get ()`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__istream_type & seekg (pos_type)`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & unget ()`

### 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>> (__streambuf_type *__sb)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`

- `__istream_type & operator>>` (long &\_\_n)
- `__istream_type & operator>>` (unsigned int &\_\_n)
- `__istream_type & operator>>` (int &\_\_n)
- `__istream_type & operator>>` (unsigned short &\_\_n)
- `__istream_type & operator>>` (short &\_\_n)
- `__istream_type & operator>>` (bool &\_\_n)
  
- `__istream_type & operator>>` (`ios_base &(*__pf)(ios_base &)`)
- `__istream_type & operator>>` (`__ios_type &(*__pf)(__ios_type &)`)
- `__istream_type & operator>>` (`__istream_type &(*__pf)(__istream_type &)`)

### 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`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits >> __num_put_type`
- `operator void * () const`
- `bool operator! () const`

### 5.373.1 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 415 of file `fstream`.

### 5.373.2 Member Typedef Documentation

**5.373.2.1** `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 71 of file `istream`.

**5.373.2.2** `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 70 of file `istream`.



**5.373.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::_num_put_type` **[inherited]**

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

**5.373.2.4** `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 419 of file `fstream`.

**5.373.2.5** `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 444 of file `ios_base.h`.

**5.373.2.6** `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 263 of file ios\_base.h.

**5.373.2.7** `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ifstream< _CharT, _Traits >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 421 of file fstream.

**5.373.2.8** `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 338 of file `ios_base.h`.

**5.373.2.9** `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_ifstream<_CharT, _Traits>::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits>](#).

Definition at line 423 of file `fstream`.

**5.373.2.10** `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 369 of file `ios_base.h`.

**5.373.2.11** `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ifstream< _CharT, _Traits >::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 422 of file `fstream`.

**5.373.2.12** `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 401 of file `ios_base.h`.

**5.373.2.13** `template<typename _CharT , typename _Traits > typedef _Traits std::basic_ifstream< _CharT, _Traits >::traits_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 420 of file `fstream`.

### 5.373.3 Member Enumeration Documentation

**5.373.3.1** `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 427 of file `ios_base.h`.

### 5.373.4 Constructor & Destructor Documentation

**5.373.4.1** `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 441 of file `fstream`.

**5.373.4.2** `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 455 of file `fstream`.

**5.373.4.3** `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 471 of file `fstream`.

**5.373.4.4** `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 486 of file fstream.

## 5.373.5 Member Function Documentation

**5.373.5.1** `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 705 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()`.

**5.373.5.2** `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 201 of file basic\_ios.h.

**5.373.5.3** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::clear(iostate __state = goodbit)  
[inline, 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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.373.5.4** `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 564 of file `fstream`.

**5.373.5.5** `template<typename _CharT, typename _Traits> basic_ios<  
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt  
(const basic_ios<_CharT, _Traits> & __rhs) [inline,  
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 62 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()`.

### 5.373.5.6 `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 180 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()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

### 5.373.5.7 `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 247 of file `basic_ios.h`.



**5.373.5.8** `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 212 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.373.5.9** `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 191 of file basic\_ios.h.

Referenced by `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()`.

**5.373.5.10** `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 380 of file `basic_ios.h`.

**5.373.5.11** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.373.5.12** `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 561 of file `ios_base.h`.

**5.373.5.13** `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

**Returns:**

The format control flags for both input and output.

## 5.373 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference 1763

Definition at line 550 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<<()`, and `std::operator>>()`.

**5.373.5.14** `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 249 of file `istream`.

**5.373.5.15** `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 366 of file `istream`.

**5.373.5.16** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get  
(__streambuf_type & __sb, char_type __delim) [inline,  
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 356 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()`.

**5.373.5.17** `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 333 of file istream.

**5.373.5.18** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::get(char_type * __s, streamsize __n, char_type __delim) [inline, 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 309 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_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.373.5.19** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::get(char_type & __c) [inline, 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 272 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()`.

**5.373.5.20** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get(void) [inline, 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 236 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()`.

**5.373.5.21** `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 406 of file `istream`.

Referenced by `std::basic_ifstream< char >::getline()`.

**5.373.5.22** `template<typename _CharT , typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::getline (char_type * __s, streamsize __n, char_type __delim) [inline, 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 400 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()`.

#### 5.373.5.23 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 694 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()`.

#### 5.373.5.24 `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 170 of file `basic_ios.h`.

#### 5.373.5.25 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim)` [`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 555 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::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()`.

#### 5.373.5.26 `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::ignore (streamsize __n) [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 493 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::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()`.

#### 5.373.5.27 `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::ignore (void) [inline, 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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.373.5.28** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file basic\_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.373.5.29** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits >::init (basic_streambuf<_CharT,  
_Traits > * __sb) [inline, 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 125 of file basic\_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.373.5.30** `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 505 of file fstream.

**5.373.5.31** `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 740 of file ios\_base.h.

**5.373.5.32** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.373.5.33** `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 546 of file `fstream`.

**5.373.5.34** `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 526 of file `fstream`.

**5.373.5.35** `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 111 of file `basic_ios.h`.

**5.373.5.36** `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 115 of file `basic_ios.h`.

**5.373.5.37** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (_streambuf_type * __sb)` [`inline`, `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 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.373.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (void *& __p) [inline, 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 215 of file istream.

**5.373.5.39** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits >::operator>> (long double & __f) [inline, 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 211 of file istream.

**5.373.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits >::operator>> (double & __f) [inline, 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 207 of file istream.

**5.373.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (float & __f) [inline, 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 203 of file istream.

**5.373.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (unsigned long long & __n) [inline, 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 198 of file istream.

**5.373.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits >::operator>> (long long & __n) [inline, 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 194 of file istream.

**5.373.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline, 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 189 of file istream.

**5.373.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline, 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,

## 5.373 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1779

- 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 185 of file `istream`.

**5.373.5.46** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned int &_n) [inline, 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 181 of file `istream`.

**5.373.5.47** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (int &_n) [inline, 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 159 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()`.

**5.373.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned short & __n) [inline, 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 174 of file istream.

**5.373.5.49** `template<typename _CharT, typename _Traits > basic_ifstream<_CharT, _Traits > & std::basic_ifstream<_CharT, _Traits >::operator>> (short & __n) [inline, 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 114 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\(\)](#).

**5.373.5.50** `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 167 of file istream.

**5.373.5.51** `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 131 of file `istream`.

**5.373.5.52** `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 124 of file `istream`.

**5.373.5.53** `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 `iomanip` header.

Definition at line 120 of file `istream`.

**5.373.5.54** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek(void) [inline, 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 620 of file `istream.tcc`.

## 5.373 std::basic\_ifstream<\_CharT, \_Traits> Class Template Reference 1783

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().

### 5.373.5.55 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 629 of file ios\_base.h.

### 5.373.5.56 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 620 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

### 5.373.5.57 template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & std::basic\_istream<\_CharT, \_Traits>::putback (char\_type \_\_c) [inline, 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 711 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

### 5.373.5.58 `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 761 of file `ios_base.h`.

### 5.373.5.59 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf(basic_streambuf<_CharT, _Traits> * __sb)` [`inline`, `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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

#### 5.373.5.60 `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 497 of file `fstream`.

#### 5.373.5.61 `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 127 of file `basic_ios.h`.

**5.373.5.62** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::read(char_type * __s, streamsize __n) [inline, 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 650 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()`.

**5.373.5.63** `template<typename _CharT, typename _Traits > streamsize std::basic_istream<_CharT, _Traits >::readsome(char_type * __s, streamsize __n) [inline, 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.

## 5.373 std::basic\_ifstream<\_CharT, \_Traits> Class Template Reference 1787

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 679 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()`.

### 5.373.5.64 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.

### 5.373.5.65 template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & std::basic\_istream<\_CharT, \_Traits>::seekg (off\_type \_\_off, ios\_base::seekdir \_\_dir) [inline, 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 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.373.5.66** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg(pos_type __pos) [inline, 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 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.373.5.67** `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 594 of file `ios_base.h`.

### 5.373.5.68 `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

### 5.373.5.69 `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 147 of file basic\_ios.h.

Referenced by `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_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>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.373.5.70** `template<typename _CharT, typename _Traits> int std::basic_istream<_CharT, _Traits>::sync(void) [inline, 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 777 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()`.

**5.373.5.71** `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>

**5.373.5.72** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits>::pos_type std::basic_ifstream<_CharT, _Traits>::tellg(void) [inline, 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 813 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()`.

**5.373.5.73** `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 297 of file `basic_ios.h`.

**5.373.5.74** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.373.5.75** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::unget  
(void) [inline, 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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sungetc()`.



**5.373.5.76** `void std::ios_base::unseff (fmtflags __mask) [inline, inherited]`

Clearing format flags.

**Parameters:**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.373.5.77** `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 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.373.5.78** `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 652 of file ios\_base.h.

**5.373.5.79** `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 643 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.373.5.80** `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.

**5.373.6 Member Data Documentation****5.373.6.1** `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\(\)](#).

## 5.373 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference 1795

Definition at line 79 of file `istream`.

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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.373.6.2 `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 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter>::do_put()`.

### 5.373.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

### 5.373.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

### 5.373.6.5 `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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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>::sync()`, `std::basic_istream<_CharT, _Traits`

>::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::basic\_ostream< \_CharT, \_Traits >::write().

#### 5.373.6.6 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 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.373.6.7 const seekdir std::ios\_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### 5.373.6.8 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 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### 5.373.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.373.6.10 const seekdir std::ios\_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc`

## 5.373 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference 1797

>::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

### 5.373.6.11 const fmtflags std::ios\_base::dec [static, inherited]

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file ios\_base.h.

### 5.373.6.12 const seekdir std::ios\_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

### 5.373.6.13 const iostate std::ios\_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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\_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 >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

### 5.373.6.14 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 350 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\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), 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()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

#### 5.373.6.15 `const fmtflags std::ios_base::fixed` [`static`, `inherited`]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

#### 5.373.6.16 `const fmtflags std::ios_base::floatfield` [`static`, `inherited`]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

#### 5.373.6.17 `const iostate std::ios_base::goodbit` [`static`, `inherited`]

Indicates all is well.

Definition at line 353 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>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### 5.373.6.18 `const fmtflags std::ios_base::hex` [`static`, `inherited`]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.373.6.19** `const openmode std::ios_base::in` [`static`, `inherited`]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `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_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()`.

**5.373.6.20** `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 280 of file `ios_base.h`.

**5.373.6.21** `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter>::do_put()`.

**5.373.6.22** `const fmtflags std::ios_base::oct` [`static`, `inherited`]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.373.6.23** `const openmode std::ios_base::out` [`static`, `inherited`]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `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()`.

#### **5.373.6.24 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 291 of file `ios_base.h`.

#### **5.373.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

#### **5.373.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file `ios_base.h`.

#### **5.373.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file `ios_base.h`.

#### **5.373.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

#### **5.373.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

#### **5.373.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.



## 5.373 std::basic\_ifstream<\_CharT, \_Traits > Class Template Reference 1801

### **5.373.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

### **5.373.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

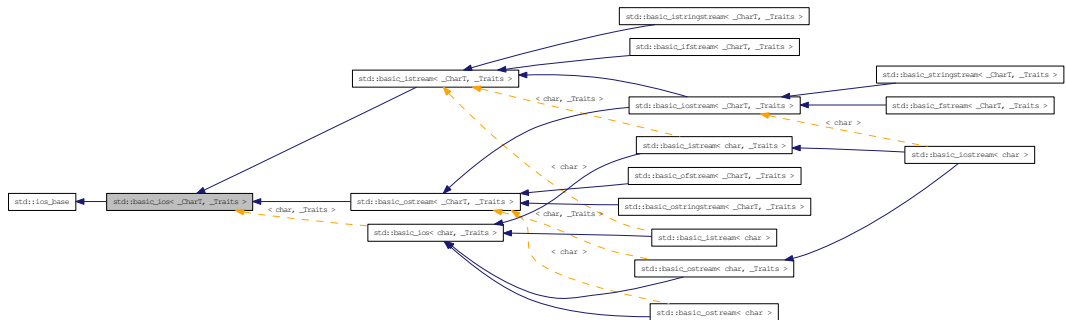
The documentation for this class was generated from the following file:

- [fstream](#)

## 5.374 `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`

### 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
- [void exceptions](#) (iostate \_\_except)
- [iostate exceptions](#) () const
- [bool fail](#) () const
- [char\\_type fill](#) (char\_type \_\_ch)
- [char\\_type fill](#) () const
- [fmtflags flags](#) (fmtflags \_\_fmtfl)
- [fmtflags flags](#) () const
- [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](#) (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](#) ([basic\\_ostream<\\_CharT, \\_Traits > \\* \\_\\_t](#) -  
\_fiestr)
- [basic\\_ostream<\\_CharT, \\_Traits > \\* tie](#) () const
- [void unsetf](#) (fmtflags \_\_mask)
- [char\\_type widen](#) (char \_\_c) const
- [streamsize width](#) (streamsize \_\_wide)
- [streamsize width](#) () 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

- [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](#)
  - [iostate](#) [\\_M\\_streambuf\\_state](#)
  - [basic\\_ostream](#)<\_CharT, \_Traits > \* [\\_M\\_tie](#)
  - [streamsize](#) [\\_M\\_width](#)
  - [\\_Words](#) \* [\\_M\\_word](#)
  - int [\\_M\\_word\\_size](#)
  - [\\_Words](#) [\\_M\\_word\\_zero](#)
- 
- typedef [ctype](#)<\_CharT > [\\_\\_ctype\\_type](#)
  - typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits > > [\\_\\_num\\_get\\_type](#)
  - typedef [num\\_put](#)<\_CharT, [ostreambuf\\_iterator](#)<\_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)
  - typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits::int\\_type](#) [int\\_type](#)
  - typedef [\\_Traits::off\\_type](#) [off\\_type](#)
  - typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - [operator void \\*](#) () const
  - bool [operator!](#) () const

### 5.374.1 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 62 of file `basic_ios.h`.

### 5.374.2 Member Typedef Documentation

**5.374.2.1** `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 82 of file `basic_ios.h`.

**5.374.2.2** `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 86 of file `basic_ios.h`.

**5.374.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [std::basic\\_ostream< char, \\_Traits >](#), and [std::basic\\_ostream< char >](#).

Definition at line 84 of file `basic_ios.h`.

#### 5.374.2.4 `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 71 of file `basic_ios.h`.

#### 5.374.2.5 `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 444 of file `ios_base.h`.

#### 5.374.2.6 `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 263 of file `ios_base.h`.

#### 5.374.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are non-standard types.

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<_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 72 of file `basic_ios.h`.

---



**5.374.2.8 `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 338 of file `ios_base.h`.

**5.374.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios<_CharT, _Traits>::off_type`**

These are non-standard types.

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`.

**5.374.2.10 `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 369 of file ios\_base.h.

#### 5.374.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type`

These are non-standard types.

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_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 73 of file basic\_ios.h.

#### 5.374.2.12 `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 401 of file ios\_base.h.

#### 5.374.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type`

These are non-standard types.

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_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 75 of file basic\_ios.h.

### 5.374.3 Member Enumeration Documentation

#### 5.374.3.1 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 427 of file ios\_base.h.

### 5.374.4 Constructor & Destructor Documentation

#### 5.374.4.1 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 260 of file basic\_ios.h.

#### 5.374.4.2 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 272 of file basic\_ios.h.

#### 5.374.4.3 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 450 of file basic\_ios.h.

### 5.374.5 Member Function Documentation

#### 5.374.5.1 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 705 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()`.

#### 5.374.5.2 `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 201 of file `basic_ios.h`.

#### 5.374.5.3 `template<typename _CharT , typename _Traits > void std::basic_ios<_CharT, _Traits >::clear (iostate __state = goodbit) [inline]`

[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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits >::rdbuf()`.

**5.374.5.4** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt(const basic_ios<_CharT, _Traits> & __rhs) [inline]`

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 62 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()`.

**5.374.5.5** `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 180 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()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.374.5.6** `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 247 of file `basic_ios.h`.

**5.374.5.7** `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 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.374.5.8** `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 191 of file basic\_ios.h.

Referenced by 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().

**5.374.5.9** `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 380 of file basic\_ios.h.

**5.374.5.10** `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 360 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt().

### 5.374.5.11 `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 561 of file `ios_base.h`.

### 5.374.5.12 `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 550 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<<()`, and `std::operator>>()`.

### 5.374.5.13 `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 694 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()`.



**5.374.5.14** `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 170 of file `basic_ios.h`.

**5.374.5.15** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue(const locale & __loc)  
[inline]`

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 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.374.5.16** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init(basic_streambuf<_CharT,  
_Traits> * __sb) [inline, protected]`

All setup is performed here. This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.374.5.17** `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 `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 740 of file `ios_base.h`.

**5.374.5.18** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.374.5.19** `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 111 of file `basic_ios.h`.

**5.374.5.20** `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 115 of file `basic_ios.h`.

**5.374.5.21** `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 629 of file `ios_base.h`.

**5.374.5.22** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.374.5.23** `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 761 of file `ios_base.h`.

**5.374.5.24** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (basic_streambuf<_CharT, _Traits > * __sb) [inline]`

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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits >::clear()`.

**5.374.5.25** `template<typename _CharT, typename _Traits>  
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,  
_Traits>::rdbuf() 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\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#), and [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 311 of file `basic_ios.h`.

Referenced by [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>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

**5.374.5.26** `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 127 of file `basic_ios.h`.

**5.374.5.27** `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.

**5.374.5.28** `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 594 of file `ios_base.h`.

**5.374.5.29** `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.374.5.30** `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 147 of file `basic_ios.h`.

Referenced by `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_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>::sync()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

**5.374.5.31** `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>

**5.374.5.32** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie (basic_ostream<_CharT, _Traits > * __tistr) [inline]`

Ties this stream to an output stream.

**Parameters:**

*tistr* 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 297 of file `basic_ios.h`.

**5.374.5.33** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.374.5.34** `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 609 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.374.5.35** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic\_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

**5.374.5.36** `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 652 of file ios\_base.h.

**5.374.5.37 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 643 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits >::copyfmt(), std::num\_put<\_CharT, \_OutIter >::do\_put(), and std::operator>>().

**5.374.5.38 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.

## 5.374.6 Member Data Documentation

**5.374.6.1 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 318 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

**5.374.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file ios\_base.h.

**5.374.6.3 `const openmode std::ios_base::ate` [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**5.374.6.4 `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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::basic_ostream<_CharT, _Traits>::write()`.

**5.374.6.5 `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 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, InIter>::do_get()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.374.6.6 `const seekdir std::ios_base::beg` [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

**5.374.6.7 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 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**5.374.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.374.6.9 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), 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().

**5.374.6.10 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file ios\_base.h.

**5.374.6.11 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.374.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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\_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 >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

**5.374.6.13 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 350 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\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), 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(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.374.6.14 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.374.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.374.6.16 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::sync()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.374.6.17 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.374.6.18 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by `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_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()`.

**5.374.6.19 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 280 of file ios\_base.h.

#### 5.374.6.20 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 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.374.6.21 const fmtflags std::ios\_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.374.6.22 const openmode std::ios\_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), 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().

#### 5.374.6.23 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 291 of file ios\_base.h.

#### 5.374.6.24 const fmtflags std::ios\_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.374.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file ios\_base.h.

**5.374.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file ios\_base.h.

**5.374.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.374.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.374.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.374.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.374.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

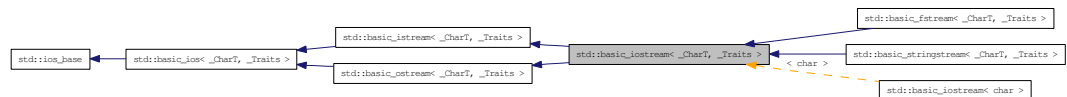


- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## 5.375 `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 `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`
- 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`

## 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
- void [exceptions](#) ([iostate](#) \_\_except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [char\\_type](#) [fill](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [\\_\\_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, [int\\_type](#) \_\_delim)
- [template](#)<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [template](#)<>  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [template](#)<>  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [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`
- `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 __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `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`

### 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.

- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### 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.

- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type &__c)`
- `int_type get ()`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__istream_type & seekg (pos_type)`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & unget ()`

### 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<< (__streambuf_type *__sb)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`

- `__ostream_type & operator<< (ios_base &(*_pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*_pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*_pf)(__ostream_type &))`

### 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>> (__streambuf_type *__sb)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (ios_base &(*_pf)(ios_base &))`
- `__istream_type & operator>> (__ios_type &(*_pf)(__ios_type &))`
- `__istream_type & operator>> (__istream_type &(*_pf)(__istream_type &))`

### 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**
- typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)
- [operator void](#) \* () const
- bool [operator!](#) () const



### 5.375.1 Detailed Description

```
template<typename _CharT, typename _Traits> 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.

Definition at line 769 of file istream.

### 5.375.2 Member Typedef Documentation

```
5.375.2.1 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 71 of file ostream.

```
5.375.2.2 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 71 of file istream.

```
5.375.2.3 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 70 of file istream.

**5.375.2.4** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits >::__num_put_type`  
**[inherited]**

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file `ostream`.

**5.375.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits >::__num_put_type`  
**[inherited]**

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits >](#), [std::basic\\_ostream<char, \\_Traits >](#), and [std::basic\\_ostream<char >](#).

Definition at line 84 of file `basic_ios.h`.

**5.375.2.6** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_iostream<_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 >](#).

Reimplemented in [std::basic\\_fstream<\\_CharT, \\_Traits >](#), and [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 777 of file `istream`.

**5.375.2.7** `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 444 of file [ios\\_base.h](#).

### **5.375.2.8 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 263 of file [ios\\_base.h](#).

**5.375.2.9** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream< \\_CharT, \\_Traits >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 778 of file `istream`.

**5.375.2.10** `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 338 of file `ios_base.h`.

**5.375.2.11** `template<typename _CharT, typename _Traits> typedef  
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream< \\_CharT, \\_Traits >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 780 of file `istream`.

**5.375.2.12** `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 369 of file `ios_base.h`.

**5.375.2.13** `template<typename _CharT, typename _Traits> typedef`  
`_Traits::pos_type std::basic_iostream< _CharT, _Traits`  
`>::pos_type`

These are non-standard types.

Reimplemented from `std::basic_istream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 779 of file `istream`.

**5.375.2.14** `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 401 of file `ios_base.h`.

**5.375.2.15** `template<typename _CharT, typename _Traits> typedef _Traits`  
`std::basic_iostream< _CharT, _Traits >::traits_type`

These are non-standard types.

Reimplemented from `std::basic_istream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 781 of file `istream`.

### 5.375.3 Member Enumeration Documentation

#### 5.375.3.1 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 427 of file `ios_base.h`.

### 5.375.4 Constructor & Destructor Documentation

#### 5.375.4.1 `template<typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>::basic_ostream (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 794 of file `istream`.

#### 5.375.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_ostream<_CharT, _Traits>::~~basic_ostream () [inline, virtual]`

Destructor does nothing.

Definition at line 801 of file `istream`.

### 5.375.5 Member Function Documentation

#### 5.375.5.1 `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 705 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().

**5.375.5.2** `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 287 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**5.375.5.3** `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 201 of file basic\_ios.h.

**5.375.5.4** `template<typename _CharT, typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)  
[inline, 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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.375.5.5** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt(const basic_ios<_CharT, _Traits> & __rhs) [inline, 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 62 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()`.

**5.375.5.6** `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 180 of file `basic_ios.h`.



## 5.375 std::basic\_istream<\_CharT, \_Traits> Class Template Reference 1849

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()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.375.5.7 `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 247 of file `basic_ios.h`.

### 5.375.5.8 `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 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

#### 5.375.5.9 `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 191 of file `basic_ios.h`.

Referenced by `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()`.

#### 5.375.5.10 `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 380 of file `basic_ios.h`.

### 5.375.5.11 `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.375.5.12 `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 561 of file `ios_base.h`.

### 5.375.5.13 `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 550 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<<()`, and `std::operator>>()`.

**5.375.5.14** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush() [inline, 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 211 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()`.

**5.375.5.15** `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 249 of file istream.

**5.375.5.16** `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 366 of file istream.

**5.375.5.17** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get(_streambuf_type & __sb, char_type __delim) [inline, 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 356 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>::putc()`.

**5.375.5.18** `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 333 of file istream.

**5.375.5.19** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::get(char_type * __s, streamsize __n, char_type __delim) [inline, 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 309 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()`.

**5.375.5.20** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c) [inline, 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 272 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().

**5.375.5.21** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (void) [inline, 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 236 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().

**5.375.5.22** `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 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

**5.375.5.23** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n, char_type __delim) [inline, 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 400 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().

#### 5.375.5.24 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 694 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().

#### 5.375.5.25 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 170 of file basic\_ios.h.

**5.375.5.26** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [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 555 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()`.

**5.375.5.27** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n) [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 493 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()`.

**5.375.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(void) [inline, 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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.375.5.29** `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue(const locale & __loc) [inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.375.5.30** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init(basic_streambuf<_CharT,  
_Traits> * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.375.5.31** `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 740 of file `ios_base.h`.

**5.375.5.32** `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.html>

Definition at line 420 of file `basic_ios.h`.

**5.375.5.33** `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 111 of file `basic_ios.h`.

**5.375.5.34** `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 115 of file `basic_ios.h`.

**5.375.5.35** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (_streambuf_type * __sb) [inline, 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 120 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()`.

**5.375.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (const void * __p) [inline, 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 225 of file ostream.

**5.375.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long double __f) [inline, 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 221 of file ostream.

**5.375.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) [inline, 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 213 of file ostream.

**5.375.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline, 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 209 of file ostream.

**5.375.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long long __n) [inline, 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 204 of file ostream.

**5.375.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, 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 200 of file ostream.

**5.375.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned int __n) [inline, 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 191 of file ostream.

**5.375.5.43** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n) [inline, 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 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.375.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned short __n) [inline, 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 180 of file ostream.

**5.375.5.45** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (short __n) [inline, 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 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.375.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (bool __n) [inline, 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 173 of file ostream.

**5.375.5.47** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long __n) [inline, 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 169 of file ostream.

**5.375.5.48** `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 165 of file ostream.

**5.375.5.49** `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 127 of file ostream.

**5.375.5.50** `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 117 of file ostream.

**5.375.5.51** `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 108 of file `ostream`.

**5.375.5.52** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>(__streambuf_type * _sb) [inline, 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 204 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()`.

**5.375.5.53** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline, 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 215 of file istream.

**5.375.5.54** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, 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 211 of file `istream`.

**5.375.555** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (double & __f) [inline, 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 207 of file `istream`.

**5.375.556** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (float & __f) [inline, 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 203 of file istream.

**5.375.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (unsigned long long & __n) [inline, 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 198 of file istream.

**5.375.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long long & __n) [inline, 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 194 of file `istream`.

**5.375.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (unsigned long & __n) [inline, 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 189 of file istream.

**5.375.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long & __n) [inline, 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 185 of file istream.

**5.375.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (unsigned int & __n) [inline, 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 181 of file `istream`.

**5.375.5.62** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>(int & __n) [inline, 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 159 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()`.

**5.375.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, 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 174 of file istream.

**5.375.5.64** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (short & __n) [inline, 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 114 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\(\)](#).

**5.375.5.65** `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 167 of file istream.

**5.375.5.66** `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 131 of file istream.

**5.375.5.67** `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 124 of file istream.

**5.375.5.68** `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 `iomanip` header.

Definition at line 120 of file istream.

**5.375.5.69** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek(void) [inline, 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 620 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()`.

**5.375.5.70** `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 629 of file ios\_base.h.



**5.375.5.71** 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 620 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

**5.375.5.72** template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::put(char\_type \_\_c) [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 149 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().

**5.375.5.73** template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & std::basic\_istream<\_CharT, \_Traits>::putback(char\_type \_\_c) [inline, 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 711 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**5.375.5.74 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 761 of file `ios_base.h`.

**5.375.5.75** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf(basic_streambuf<_CharT, _Traits> * __sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.375.5.76** `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_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `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>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.375.5.77** `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 127 of file `basic_ios.h`.

**5.375.5.78** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read  
(char_type * __s, streamsize __n) [inline, 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 650 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_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.375.5.79** `template<typename _CharT, typename _Traits> streamsize`  
`std::basic_istream<_CharT, _Traits>::readsome(char_type * __s,`  
`streamsize __n) [inline, 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 679 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_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.375.5.80** `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.

**5.375.5.81** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inline, 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 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.375.5.82** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inline, 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 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.375.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir)` [`inline`, `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 290 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()`.

**5.375.5.84** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (pos_type __pos) [inline, 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 258 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()`.

**5.375.5.85** `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 594 of file ios\_base.h.

**5.375.5.86** `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 577 of file ios\_base.h.

Referenced by [std::boolalpha\(\)](#), [std::dec\(\)](#), [std::fixed\(\)](#), [std::hex\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), [std::oct\(\)](#), [std::right\(\)](#), [std::scientific\(\)](#), [std::showbase\(\)](#), [std::showpoint\(\)](#), [std::showpos\(\)](#), [std::skipws\(\)](#), [std::unitbuf\(\)](#), and [std::uppercase\(\)](#).

**5.375.5.87** `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 147 of file basic\_ios.h.

Referenced by [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\\_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 >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**5.375.5.88** `template<typename _CharT, typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync (void) [inline,  
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 777 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()`.

### 5.375.5.89 `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>

### 5.375.5.90 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (void) [inline, 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 813 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()`.

**5.375.5.91** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inline, 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 237 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()`.

**5.375.5.92** `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 297 of file `basic_ios.h`.

**5.375.5.93** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.375.5.94** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
(void) [inline, 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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.375.5.95** 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 609 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.375.5.96** 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 notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic\_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

**5.375.5.97** 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 652 of file ios\_base.h.

**5.375.5.98** `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 643 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.375.5.99** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::write (const char_type * __s, streamsize __n) [inline, 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.

## 5.375 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 1895

Definition at line 183 of file `ostream.tcc`.

References `std::basic_ostream<_CharT, _Traits>::_M_write()`, and `std::ios_base::badbit`.

### 5.375.5.100 `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.

## 5.375.6 Member Data Documentation

### 5.375.6.1 `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 79 of file `istream`.

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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.375.6.2 `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 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.375.6.3 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file ios\_base.h.

**5.375.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.375.6.5 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 342 of file ios\_base.h.

Referenced by 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::operator<<(), 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 >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::basic\_ostream< \_CharT, \_Traits >::write().

**5.375.6.6 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 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.375.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.



Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### **5.375.6.8 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 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### **5.375.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

#### **5.375.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), 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().

#### **5.375.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in `decimal` base.

Definition at line 269 of file ios\_base.h.

#### **5.375.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 5.375.6.13 `const iostate std::ios_base::eofbit` [`static`, `inherited`]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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 >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

#### 5.375.6.14 `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 350 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_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `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()`, and `std::basic_ostream< _CharT, _Traits >::seekp()`.

#### 5.375.6.15 `const fmtflags std::ios_base::fixed` [`static`, `inherited`]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

#### 5.375.6.16 `const fmtflags std::ios_base::floatfield` [`static`, `inherited`]

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

**5.375.6.17 `const ios_base::goodbit` [static, inherited]**

Indicates all is well.

Definition at line 353 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>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.375.6.18 `const fmtflags std::ios_base::hex` [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.375.6.19 `const ios_base::in` [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `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_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()`.

**5.375.6.20 `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 280 of file `ios_base.h`.

#### **5.375.6.21 `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

#### **5.375.6.22 `const fmtflags std::ios_base::oct` [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### **5.375.6.23 `const openmode std::ios_base::out` [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf<_CharT, _Traits >::overflow()`, `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 >::xspn()`.

#### **5.375.6.24 `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 291 of file `ios_base.h`.

#### **5.375.6.25 `const fmtflags std::ios_base::scientific` [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

---

**5.375.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file ios\_base.h.

**5.375.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file ios\_base.h.

**5.375.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.375.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.375.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.375.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.375.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

- [istream](#)



- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**
- typedef `_Traits` [traits\\_type](#)

## Public Member Functions

- [basic\\_istream](#) (`__streambuf_type *__sb`)
- virtual `~basic_istream` ()
- const `locale & _M_getloc` () const
- void `_M_setstate` (`iosstate __state`)
- bool `bad` () const
- void `clear` (`iosstate __state=goodbit`)
- `basic_ios & copyfmt` (const `basic_ios &__rhs`)
- bool `eof` () const
- void `exceptions` (`iosstate __except`)
- `iosstate exceptions` () const
- bool `fail` () const
- `char_type fill` (`char_type __ch`)
- `char_type fill` () const
- `fmtflags flags` (`fmtflags __fmtfl`)
- `fmtflags flags` () const
- `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`, `int_type __delim`)
- template<>  
`basic_istream< wchar_t > & ignore` (`streamsize __n`)
- template<>  
`basic_istream< char > & ignore` (`streamsize __n`, `int_type __delim`)
- template<>  
`basic_istream< char > & ignore` (`streamsize __n`)
- `locale imbue` (const `locale &__loc`)
- long & `isword` (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
- `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 (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

### 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.

- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type & __c)`
- `int_type get ()`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type * __s, streamsize __n)`

- [streamsize](#) [readsome](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [seekg](#) ([off\\_type](#), [ios\\_base::seekdir](#))
- [\\_\\_istream\\_type](#) & [seekg](#) ([pos\\_type](#))
- [int](#) [sync](#) ()
- [pos\\_type](#) [tellg](#) ()
- [\\_\\_istream\\_type](#) & [unget](#) ()

### 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>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([void](#) \*&\_\_p)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([long double](#) &\_\_f)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([double](#) &\_\_f)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([float](#) &\_\_f)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned long long](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([long long](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned long](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([long](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned int](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([int](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned short](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([short](#) &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([bool](#) &\_\_n)
- 
- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) &(\*\_\_pf)([\\_\\_istream\\_type](#) &))

### 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`
- `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`
- typedef `num_put`< `_CharT`, `ostreambuf_iterator`< `_CharT`, `_Traits` > > `__num_put_type`
- `operator void` \* () const
- bool `operator!` () const

### 5.376.1 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 55 of file `istream`.

### 5.376.2 Member Typedef Documentation

```
5.376.2.1 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 71 of file `istream`.

```
5.376.2.2 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 70 of file `istream`.

```
5.376.2.3 template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits >::__num_put_type
[inherited]
```

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

#### 5.376.2.4 `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 >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits, _Alloc >`, `std::basic_ostream< _CharT, _Traits, _Alloc >`, and `std::basic_iostream< char >`.

Definition at line 59 of file `istream`.

#### 5.376.2.5 `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 444 of file `ios_base.h`.

#### 5.376.2.6 `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 263 of file `ios_base.h`.

#### 5.376.2.7 `template<typename _CharT, typename _Traits> typedef` `_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostream<_CharT, _Traits, _Alloc>`, and `std::basic_iostream<char>`.

Definition at line 60 of file `istream`.

#### 5.376.2.8 `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 338 of file ios\_base.h.

#### 5.376.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_istream< _CharT, _Traits >::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >](#), and [std::basic\\_ostream< char >](#).

Definition at line 62 of file istream.

#### 5.376.2.10 `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 369 of file ios\_base.h.

#### 5.376.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).



## 5.376 std::basic\_istream< \_CharT, \_Traits > Class Template Reference 1913

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [std::basic\\_istringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), and [std::basic\\_istream< char >](#).

Definition at line 61 of file `istream`.

### 5.376.2.12 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 401 of file `ios_base.h`.

### 5.376.2.13 template<typename \_CharT, typename \_Traits> typedef \_Traits std::basic\_istream< \_CharT, \_Traits >::traits\_type

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [std::basic\\_istringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), and [std::basic\\_istream< char >](#).

Definition at line 63 of file `istream`.

## 5.376.3 Member Enumeration Documentation

### 5.376.3.1 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 427 of file `ios_base.h`.

## 5.376.4 Constructor & Destructor Documentation

**5.376.4.1** `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 91 of file istream.

**5.376.4.2** `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 101 of file istream.

## 5.376.5 Member Function Documentation

**5.376.5.1** `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 705 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()`.

**5.376.5.2** `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 201 of file basic\_ios.h.

**5.376.5.3** `template<typename _CharT , typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)  
[inline, 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 40 of file basic\_ios.tcc.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**5.376.5.4** `template<typename _CharT , typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt  
(const basic_ios< _CharT, _Traits > & __rhs) [inline,  
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 62 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\(\)](#).

### 5.376.5.5 `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 180 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()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

### 5.376.5.6 `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 247 of file `basic_ios.h`.

**5.376.5.7** `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 212 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.376.5.8** `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 191 of file basic\_ios.h.

Referenced by `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()`.

**5.376.5.9** `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 380 of file `basic_ios.h`.

**5.376.5.10** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.376.5.11** `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 561 of file `ios_base.h`.

**5.376.5.12** `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

**Returns:**

The format control flags for both input and output.

## 5.376 std::basic\_istream< \_CharT, \_Traits > Class Template Reference 1919

Definition at line 550 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<<(), and std::operator>>().

### 5.376.5.13 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 249 of file istream.

### 5.376.5.14 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 366 of file istream.

### 5.376.5.15 template<typename \_CharT , typename \_Traits > basic\_istream< \_CharT, \_Traits > & std::basic\_istream< \_CharT, \_Traits >::get (\_\_streambuf\_type & \_\_sb, char\_type \_\_delim) [inline]

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 356 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()`.

**5.376.5.16** `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 333 of file istream.

---



**5.376.5.17** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim) [inline]`

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 309 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()`.

**5.376.5.18** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c) [inline]`

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 272 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()`.

**5.376.5.19** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get(void) [inline]`

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 236 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()`.

**5.376.5.20** `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 406 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**5.376.5.21** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline(char_type * __s, streamsize __n, char_type __delim) [inline]`

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 400 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()`.

#### 5.376.5.22 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 694 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()`.

#### 5.376.5.23 `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 170 of file `basic_ios.h`.

#### 5.376.5.24 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim)` [`inline`]

Extraction into another streambuf.

##### Parameters:

*sb* A streambuf in which to store data.

**Returns:**

`*this`

Returns `get(sb,widen('\n'))`.

Definition at line 555 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()`.

#### 5.376.5.25 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n) [inline]`

Extraction into another streambuf.

**Parameters:**

*sb* A streambuf in which to store data.

**Returns:**

`*this`

Returns `get(sb,widen('\n'))`.

Definition at line 493 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()`.

#### 5.376.5.26 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(void) [inline]`

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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.376.5.27** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue(const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.376.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT,  
_Traits > * __sb) [inline, 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 125 of file basic\_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.376.5.29** `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 740 of file `ios_base.h`.

**5.376.5.30** `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, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.376.5.31** `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 111 of file `basic_ios.h`.

**5.376.5.32** `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 115 of file `basic_ios.h`.

**5.376.5.33** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (_streambuf_type * __sb)` [`inline`]

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 204 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\(\)](#).

**5.376.5.34** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& _p) [inline]`

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 215 of file istream.

**5.376.5.35** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & _f) [inline]`

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 211 of file istream.

**5.376.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline]`

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 207 of file istream.

**5.376.5.37** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline]`

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 203 of file istream.

**5.376.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline]`

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 198 of file istream.

**5.376.5.39** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline]`

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 194 of file istream.

**5.376.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline]`

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 189 of file istream.

#### 5.376.5.41 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n)` [`inline`]

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 185 of file istream.

**5.376.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline]`

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 181 of file istream.

**5.376.5.43** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inline]`

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 159 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\(\)](#).

**5.376.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline]`

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 174 of file istream.

**5.376.5.45** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (short & __n) [inline]`

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 114 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()`.

**5.376.5.46** `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 167 of file istream.

**5.376.5.47** `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 131 of file `istream`.

**5.376.5.48** `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 124 of file `istream`.

**5.376.5.49** `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 `iomanip` header.

Definition at line 120 of file `istream`.

**5.376.5.50** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek(void) [inline]`

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 620 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()`.

**5.376.5.51** `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 629 of file `ios_base.h`.

**5.376.5.52** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.376.5.53** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inline]`

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 711 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::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**5.376.5.54 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 761 of file `ios_base.h`.

**5.376.5.55 template<typename \_CharT, typename \_Traits> basic\_streambuf< \_CharT, \_Traits > \* std::basic\_ios< \_CharT, \_Traits >::rdbuf (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

#### 5.376.5.56 `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_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `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>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.376.5.57** `template<typename _CharT, typename _Traits> istate  
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 127 of file `basic_ios.h`.

**5.376.5.58** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read  
(char_type * __s, streamsize __n) [inline]`

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 650 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\(\)](#).

**5.376.5.59** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome(char_type * __s,  
streamsize __n) [inline]`

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 679 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()`.

**5.376.5.60** `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.

**5.376.5.61** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inline]`

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 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.376.5.62** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inline]`

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 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

#### 5.376.5.63 `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 594 of file ios\_base.h.

#### 5.376.5.64 `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 577 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.



**5.376.5.65** `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 147 of file `basic_ios.h`.

Referenced by `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_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 >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.376.5.66** `template<typename _CharT , typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync (void) [inline]`

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 777 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()`.

#### 5.376.5.67 `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>

#### 5.376.5.68 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (void)` [`inline`]

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 813 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()`.

**5.376.5.69** `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 297 of file `basic_ios.h`.

**5.376.5.70** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.376.5.71** `template<typename _CharT, typename _Traits > basic_istream<  
 _CharT, _Traits > & std::basic_istream<_CharT, _Traits >::unget  
 (void) [inline]`

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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.376.5.72** `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 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.376.5.73** `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)
```

## 5.376 std::basic\_istream< \_CharT, \_Traits > Class Template Reference 1949

Additional information notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic\_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

### 5.376.5.74 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 652 of file ios\_base.h.

### 5.376.5.75 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 643 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::operator>>().

### 5.376.5.76 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.

## 5.376.6 Member Data Documentation

### 5.376.6.1 `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 79 of file `istream`.

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 >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

### 5.376.6.2 `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 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

### 5.376.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

### 5.376.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### 5.376.6.5 `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).

## 5.376 std::basic\_istream< \_CharT, \_Traits > Class Template Reference 1951

Definition at line 342 of file ios\_base.h.

Referenced by 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::operator<<(), 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 >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::basic\_ostream< \_CharT, \_Traits >::write().

### 5.376.6.6 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 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

### 5.376.6.7 const seekdir std::ios\_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

### 5.376.6.8 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 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

### 5.376.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### **5.376.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `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()`.

#### **5.376.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file `ios_base.h`.

#### **5.376.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### **5.376.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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 >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.



**5.376.6.14 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 350 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\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), 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(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.376.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.376.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.376.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.376.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.376.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by 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\_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().

**5.376.6.20 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 280 of file ios\_base.h.

**5.376.6.21 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 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.376.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.376.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `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()`.

**5.376.6.24 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 291 of file `ios_base.h`.

**5.376.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.376.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file `ios_base.h`.

**5.376.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file `ios_base.h`.

**5.376.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

**5.376.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.376.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.376.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.376.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 5.377 `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

#### 5.377.1 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 633 of file `istream`.

#### 5.377.2 Member Typedef Documentation

```
5.377.2.1 template<typename _CharT, typename _Traits> typedef _Traits
std::basic_istream< _CharT, _Traits >::sentry::traits_type
```

Easy access to dependant types.

Definition at line 640 of file `istream`.

### 5.377.3 Constructor & Destructor Documentation

**5.377.3.1** `template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits >::sentry::sentry (basic_istream< _CharT, _Traits > & __is, bool __noskipws = false) [inline, 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 47 of file `istream.tcc`.

References `std::__ctype_abstract_base< _CharT >::is()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::skipws()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

### 5.377.4 Member Function Documentation

**5.377.4.1** `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 681 of file istream.

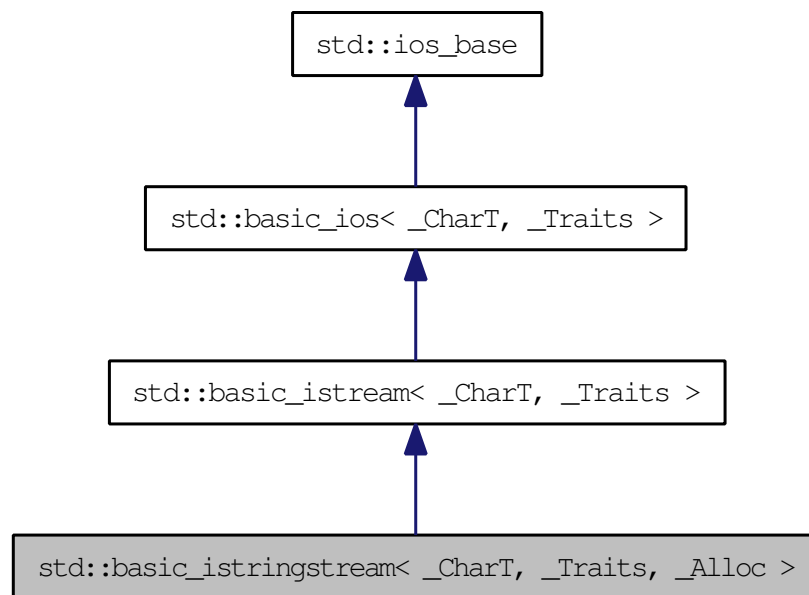
The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 5.378 `std::basic_istreamstream< _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_istreamstream< _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](#)

## Public Member Functions

- [basic\\_istream](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) ([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
- void [exceptions](#) ([iostate](#) \_\_except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [char\\_type](#) [fill](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [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, int_type __delim)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `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)`
- `__stringbuf_type * rdbuf () const`
- `iosate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `void setstate (iosate __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`

### 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.

- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type &__c)`
- `int_type get ()`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__istream_type & seekg (pos_type)`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & unget ()`

### 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>> (__streambuf_type *__sb)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (bool &__n)`

- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`

### 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`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __num_put_type`
- `operator void * () const`
- `bool operator! () const`

### 5.378.1 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 256 of file `sstream`.

### 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 71 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 70 of file `istream`.

## 5.378 `std::basic_istringstream<_CharT, _Traits, _Alloc >` Class Template Reference 1967

---

**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]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in `std::basic_ostream<_CharT, _Traits >`, `std::basic_ostream<char, _Traits >`, and `std::basic_ostream<char >`.

Definition at line 84 of file `basic_ios.h`.

**5.378.2.4** `template<typename _CharT, typename _Traits, typename _Alloc> typedef _CharT std::basic_istringstream<_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 260 of file `sstream`.

**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 444 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 263 of file ios\_base.h.

**5.378.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::int_type std::basic_istringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 265 of file sstream.



## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1969

---

### 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 338 of file `ios_base.h`.

### 5.378.2.9 `template<typename _CharT, typename _Traits, typename _Alloc> typedef traits_type::off_type std::basic_istream<_CharT, _Traits, _Alloc >::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits >](#).

Definition at line 267 of file `sstream`.

### 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`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

**5.378.2.11** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::pos_type std::basic_istream< _CharT,  
_Traits, _Alloc >::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 266 of file `sstream`.

**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 401 of file `ios_base.h`.

**5.378.2.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _Traits std::basic_istream< _CharT, _Traits, _Alloc  
>::traits_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 261 of file `sstream`.

### 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 427 of file `ios_base.h`.

## 5.378.4 Constructor & Destructor Documentation

**5.378.4.1** `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 292 of file `sstream`.

**5.378.4.2** `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 310 of file `sstream`.

**5.378.4.3** `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 321 of file sstream.

## 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 705 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 201 of file basic\_ios.h.

**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1973**

---

**5.378.5.3** `template<typename _CharT, typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)  
[inline, 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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.378.5.4** `template<typename _CharT, typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt  
(const basic_ios< _CharT, _Traits > & __rhs) [inline,  
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 62 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()`.

**5.378.5.5** `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 180 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()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

#### 5.378.5.6 `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 247 of file `basic_ios.h`.

**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1975**

---

**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 212 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.378.5.8** `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 191 of file basic\_ios.h.

Referenced by `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()`.

**5.378.5.9** `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 380 of file `basic_ios.h`.

**5.378.5.10** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.378.5.11** `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 561 of file `ios_base.h`.

**5.378.5.12** `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

**Returns:**

The format control flags for both input and output.



## 5.378 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1977

---

Definition at line 550 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<<()`, and `std::operator>>()`.

### 5.378.5.13 `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 249 of file `istream`.

### 5.378.5.14 `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 366 of file `istream`.

### 5.378.5.15 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get(__streambuf_type & __sb, char_type __delim)` [`inline`, `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 356 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()`.

**5.378.5.16** `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 333 of file istream.

**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1979**

---

**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) [inline, 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 309 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(char_type & __c) [inline, 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 272 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()`.

**5.378.5.19** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get(void) [inline, 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 236 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()`.

**5.378.5.20** `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.

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

1981

*n* Maximum number of characters to extract.

### Returns:

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char >::getline()`.

```
5.378.5.21 template<typename _CharT , typename _Traits > basic_istream<
 _CharT, _Traits > & std::basic_istream< _CharT, _Traits
 >::getline (char_type * __s, streamsize __n, char_type __delim)
 [inline, 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 400 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()`.

#### 5.378.5.22 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 694 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()`.

#### 5.378.5.23 `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 170 of file `basic_ios.h`.

#### 5.378.5.24 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim)` [`inline`, `inherited`]

Extraction into another streambuf.

##### Parameters:

*sb* A streambuf in which to store data.

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

1983

### Returns:

`*this`

Returns `get(sb, widen('\n'))`.

Definition at line 555 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()`.

### 5.378.5.25 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::ignore(streamsize __n) [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 493 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()`.

### 5.378.5.26 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::ignore(void) [inline, 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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.378.5.27** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue(const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.



**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1985**

---

**5.378.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT,  
_Traits > * __sb) [inline, 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 125 of file basic\_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.378.5.29** `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 740 of file `ios_base.h`.

**5.378.5.30** `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, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.378.5.31** `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 111 of file `basic_ios.h`.

**5.378.5.32** `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 115 of file `basic_ios.h`.

**5.378.5.33** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (_streambuf_type * __sb)` [`inline`, `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,

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1987

- 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 204 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()`.

### 5.378.5.34 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (void *& _p)` [`inline`, `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 215 of file `istream`.

### 5.378.5.35 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long double & _f)` [`inline`, `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 211 of file istream.

**5.378.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline, 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 207 of file istream.

**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1989**

---

**5.378.5.37** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, 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 203 of file istream.

**5.378.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, 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 198 of file istream.

**5.378.5.39** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline, 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 194 of file istream.

**5.378.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline, inherited]`

Extracting into another streambuf.

**Parameters:**

*sb* A pointer to a streambuf

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1991

---

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 189 of file `istream`.

### 5.378.5.41 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long & __n)` [`inline`, `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 185 of file `istream`.

**5.378.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, 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 181 of file istream.

**5.378.5.43** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inline, 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,



**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1993**

---

- 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 159 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()`.

**5.378.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, 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 174 of file istream.

**5.378.5.45** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (short & __n) [inline, 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 114 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>> (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 167 of file istream.

**5.378.5.47** `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

## 5.378 `std::basic_istringstream<_CharT, _Traits, _Alloc >` Class Template Reference 1995

---

the `iomanip` header.

Definition at line 131 of file `istream`.

**5.378.5.48** `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 124 of file `istream`.

**5.378.5.49** `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 120 of file `istream`.

**5.378.5.50** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits >::int_type std::basic_istream<_CharT, _Traits >::peek(void) [inline, 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 620 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()`.

**5.378.5.51** `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 629 of file `ios_base.h`.

**5.378.5.52** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.378.5.53** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::putback (char_type __c) [inline, 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.

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1997

---

### Note:

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 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::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

### 5.378.5.54 `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 761 of file `ios_base.h`.

### 5.378.5.55 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf(basic_streambuf<_CharT, _Traits > * __sb)` [`inline`, `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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits >::clear()`.

**5.378.5.56** `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 332 of file `sstream`.

**5.378.5.57** `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 127 of file `basic_ios.h`.

**5.378 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1999**

---

**5.378.5.58** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (char_type * __s, streamsize __n) [inline, 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 650 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()`.

**5.378.5.59** `template<typename _CharT, typename _Traits > streamsize std::basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n) [inline, 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 679 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()`.

#### 5.378.5.60 `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.

#### 5.378.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg(off_type __off, ios_base::seekdir __dir) [inline, inherited]`

Changing the current read position.

##### Parameters:

- `off` A file offset object.
- `dir` The direction in which to seek.

##### Returns:

`*this`



## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 2001

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 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.378.5.62 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::seekg(pos_type __pos)` [`inline, 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 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.378.5.63 `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 594 of file `ios_base.h`.

#### 5.378.5.64 `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

#### 5.378.5.65 `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.

## 5.378 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2003

Definition at line 147 of file `basic_ios.h`.

Referenced by `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_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>::sync()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

**5.378.5.66** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_istringstream<_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 350 of file `sstream`.

**5.378.5.67** `template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_istringstream<_CharT, _Traits, _Alloc>::str() const [inline]`

Copying out the string buffer.

### Returns:

`rdbuf() -> str()`

Definition at line 340 of file `sstream`.

**5.378.5.68** `template<typename _CharT, typename _Traits> int std::basic_istream<_CharT, _Traits>::sync(void) [inline, 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 777 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()`.

### 5.378.5.69 `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>

### 5.378.5.70 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (void) [inline, inherited]`

Getting the current read position.

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

2005

### 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 813 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()`.

**5.378.5.71** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie(basic_ostream<_CharT, _Traits > * __tistr) [inline,  
inherited]`

Ties this stream to an output stream.

### Parameters:

*tistr* 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 297 of file `basic_ios.h`.

**5.378.5.72** `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 285 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.378.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget(void) [inline, 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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.378.5.74** `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 609 of file ios\_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.378 std::basic\_istream<\_CharT, \_Traits, \_Alloc > Class Template Reference** 2007

---

**5.378.5.75** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.378.5.76** `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 652 of file `ios_base.h`.

**5.378.5.77** `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 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator<>>()`.

#### 5.378.5.78 `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.

### 5.378.6 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 79 of file `istream`.

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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### 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 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.



## 5.378 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2009

---

### 5.378.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 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 375 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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::basic_ostream<_CharT, _Traits>::write()`.

### 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 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, InIter>::do_get()`, 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 404 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 380 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 266 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### **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 407 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `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 269 of file ios\_base.h.

#### **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 410 of file ios\_base.h.

## 5.378 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 2011

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

### 5.378.6.13 `const ios_base::eofbit` [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::ws()`.

### 5.378.6.14 `const 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 350 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_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `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()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

### 5.378.6.15 `const fmtflags std::ios_base::fixed` [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

### 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 324 of file `ios_base.h`.

**5.378.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::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 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, 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 383 of file ios\_base.h.

Referenced by `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_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.

## 5.378 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 2013

---

Definition at line 280 of file `ios_base.h`.

### 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 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

### 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 287 of file `ios_base.h`.

Referenced by `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 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `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 291 of file `ios_base.h`.

### 5.378.6.25 `const fmtflags std::ios_base::scientific` [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**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 298 of file ios\_base.h.

**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 302 of file ios\_base.h.

**5.378.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.378.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.378.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 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 311 of file ios\_base.h.

**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 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

**5.378 std::basic\_istream<\_CharT, \_Traits, \_Alloc > Class Template  
Reference**

---

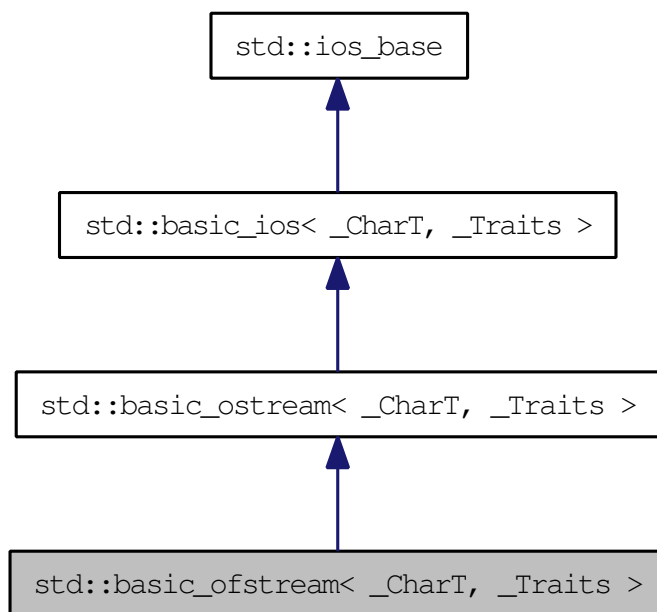
**2015**

- [sstream](#)

## 5.379 `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](#)

## Public Member Functions

- [basic\\_ofstream](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- [basic\\_ofstream](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- [basic\\_ofstream](#) ()
- [~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
- void [exceptions](#) ([iostate](#) \_\_except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [char\\_type](#) [fill](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- bool [is\\_open](#) () const
- bool [is\\_open](#) ()

- long & `iword` (int \_\_ix)
- char `narrow` (char\_type \_\_c, char \_\_dfault) const
- void `open` (const std::string &\_\_s, ios\_base::openmode \_\_mode=ios\_base::out|ios\_base::trunc)
- void `open` (const char \*\_\_s, ios\_base::openmode \_\_mode=ios\_base::out|ios\_base::trunc)
- streamsize `precision` (streamsize \_\_prec)
- `precision` () const
- 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` (off\_type, ios\_base::seekdir)
- `__ostream_type` & `seekp` (pos\_type)
- `fmtflags` `setf` (fmtflags \_\_fmtfl, fmtflags \_\_mask)
- `fmtflags` `setf` (fmtflags \_\_fmtfl)
- 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)
- `width` () const

### 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.

- void `_M_write` (const char\_type \*\_\_s, streamsize \_\_n)
- `__ostream_type` & `put` (char\_type \_\_c)
- `__ostream_type` & `write` (const char\_type \*\_\_s, streamsize \_\_n)

### 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<< (__streambuf_type *__sb)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`
  
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`

### 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`
- `typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> __num_get_type`
- `operator void * () const`
- `bool operator! () const`

### 5.379.1 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 583 of file `fstream`.

## 5.379.2 Member Typedef Documentation

**5.379.2.1** `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 71 of file ostream.

**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 [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 86 of file basic\_ios.h.

**5.379.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream< _CharT, _Traits >::_num_put_type [inherited]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.379.2.4** `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 587 of file fstream.

**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 444 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 263 of file ios\_base.h.

**5.379.2.7** `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ofstream< _CharT, _Traits >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 589 of file fstream.

**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 338 of file ios\_base.h.

**5.379.2.9** `template<typename _CharT , typename _Traits > typedef traits_type::off_type std::basic_ofstream< _CharT, _Traits >::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 591 of file fstream.



**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 369 of file `ios_base.h`.

**5.379.2.11** `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_ofstream<_CharT, _Traits>::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 590 of file `fstream`.

**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 401 of file `ios_base.h`.

**5.379.2.13** `template<typename _CharT, typename _Traits > typedef _Traits  
std::basic_ofstream< _CharT, _Traits >::traits_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 588 of file `fstream`.

### 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 427 of file `ios_base.h`.

### 5.379.4 Constructor & Destructor Documentation

**5.379.4.1** `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 609 of file `fstream`.

**5.379.4.2** `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 624 of file `fstream`.

```
5.379.4.3 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 642 of file `fstream`.

```
5.379.4.4 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 657 of file `fstream`.

## 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 705 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> 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 287 of file ostream.

Referenced by `std::basic_ostream< _CharT, _Traits >::write()`.

**5.379.5.3** `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 201 of file basic\_ios.h.

**5.379.5.4** `template<typename _CharT , typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)  
[inline, inherited]`

[Re]sets the error state.

**Parameters:**

*state* The new state flag(s) to [set](#).

## 5.379 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 2029

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

### 5.379.5.5 `template<typename _CharT, typename _Traits> void std::basic_ofstream<_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 737 of file `fstream`.

### 5.379.5.6 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt(const basic_ios<_CharT, _Traits> & __rhs)` [inline, 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 `pwd` 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 62 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()`.

### 5.379.5.7 `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 180 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()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.379.5.8** `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 247 of file `basic_ios.h`.

**5.379.5.9** `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 212 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt().

**5.379.5.10** `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 191 of file basic\_ios.h.

Referenced by 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().

**5.379.5.11** `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 380 of file basic\_ios.h.

**5.379.5.12** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.379.5.13** `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 561 of file `ios_base.h`.

**5.379.5.14** `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 550 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<<()`, and `std::operator>>()`.



**5.379.5.15** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::flush() [inline, 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 211 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()`.

**5.379.5.16** `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 694 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()`.

**5.379.5.17** `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 170 of file `basic_ios.h`.

**5.379.5.18** `template<typename _CharT , typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.379.5.19** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT,  
_Traits > * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.379.5.20** `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 676 of file `fstream`.

**5.379.5.21 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 `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 740 of file `ios_base.h`.

**5.379.5.22 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 notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.379.5.23** `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 718 of file `fstream`.

**5.379.5.24** `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 697 of file `fstream`.

**5.379.5.25** `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 111 of file `basic_ios.h`.

**5.379.5.26** `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 115 of file `basic_ios.h`.

**5.379.5.27** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits  
>::operator<<(_streambuf_type * __sb) [inline,  
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 120 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()`.

**5.379.5.28** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<(const void *  
__p) [inline, 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 225 of file ostream.

**5.379.5.29** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long double __f) [inline, 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 221 of file ostream.

```
5.379.5.30 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream<_CharT, _Traits >::operator<< (float __f)
[inline, 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 213 of file ostream.

```
5.379.5.31 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream<_CharT, _Traits >::operator<< (double __f)
[inline, 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 209 of file ostream.

**5.379.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline, 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 204 of file ostream.



**5.379.5.33** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<< (long long __n) [inline, 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 200 of file ostream.

**5.379.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<< (unsigned int __n) [inline, 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 191 of file ostream.

**5.379.5.35** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (int __n) [inline, 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 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.379.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned short __n) [inline, 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 180 of file ostream.

```
5.379.5.37 template<typename _CharT , typename _Traits > basic_ostream<
 _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits
 >::operator<< (short __n) [inline, 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 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.379.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline, 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 173 of file ostream.

**5.379.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline, 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 169 of file ostream.

**5.379.5.40** `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 165 of file ostream.

**5.379.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_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 127 of file ostream.

**5.379.5.42** `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 117 of file `ostream`.

**5.379.5.43** `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 108 of file `ostream`.

**5.379.5.44** `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 629 of file `ios_base.h`.

**5.379.5.45** `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.

## 5.379 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2047

Definition at line 620 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

**5.379.5.46** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::put(char_type __c) [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 149 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().

**5.379.5.47** `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 761 of file ios\_base.h.

**5.379.5.48** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (basic_streambuf<_CharT, _Traits > * __sb) [inline, 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 52 of file basic\_ios.tcc.

References `std::basic_ios<_CharT, _Traits >::clear()`.

**5.379.5.49** `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 668 of file fstream.



**5.379.5.50** `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 127 of file `basic_ios.h`.

**5.379.5.51** `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.

**5.379.5.52** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp  
(off_type __off, ios_base::seekdir __dir) [inline, 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 290 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()`.

**5.379.5.53** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::seekp(pos_type __pos) [inline, 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 258 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()`.

**5.379.5.54** `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 594 of file ios\_base.h.

**5.379.5.55** `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 577 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.379.5.56** `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 147 of file basic\_ios.h.

Referenced by `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_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 >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.379.5.57** `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>

**5.379.5.58** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inline, 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 237 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()`.

**5.379.5.59** `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 297 of file `basic_ios.h`.

**5.379.5.60** `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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.379.5.61** `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 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.379.5.62** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

### 5.379.5.63 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 652 of file `ios_base.h`.

### 5.379.5.64 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 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.379.5.65** `template<typename _CharT, typename _Traits > basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inline, 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 183 of file ostream.tcc.

References `std::basic_ofstream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.379.5.66** `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 `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 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

### 5.379.6.2 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 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 375 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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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 >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, `std::basic_istream<_CharT, _Traits >::unget()`, and `std::basic_ostream<_CharT, _Traits >::write()`.



**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 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, 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 404 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 380 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 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**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 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `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 269 of file ios\_base.h.

**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 410 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 345 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_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 >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, 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 350 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_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `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()`, and `std::basic_ostream<_CharT, _Traits >::seekp()`.

**5.379.6.14** `const fmtflags std::ios_base::fixed` [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

**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 324 of file `ios_base.h`.

**5.379.6.16** `const iostate std::ios_base::goodbit` [static, inherited]

Indicates all is well.

Definition at line 353 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>::sync()`, and `std::basic_istream<_CharT, _Traits>::unset()`.

**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 275 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ofstream<_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 383 of file `ios_base.h`.

Referenced by `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_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 280 of file `ios_base.h`.

#### 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 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

#### 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 287 of file `ios_base.h`.

Referenced by `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 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `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 291 of file ios\_base.h.

**5.379.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**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 298 of file ios\_base.h.

**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 302 of file ios\_base.h.

**5.379.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.379.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.379.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 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 311 of file `ios_base.h`.

**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 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

- [fstream](#)



- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**
- typedef `_Traits` [traits\\_type](#)

## Public Member Functions

- [basic\\_ostream](#) (`__streambuf_type *__sb`)
- virtual [~basic\\_ostream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) (`iosstate __state`)
- bool [bad](#) () const
- void [clear](#) (`iosstate __state=goodbit`)
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & `__rhs`)
- bool [eof](#) () const
- void [exceptions](#) (`iosstate __except`)
- `iosstate` [exceptions](#) () const
- bool [fail](#) () const
- `char_type` [fill](#) (`char_type __ch`)
- `char_type` [fill](#) () const
- `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](#) (`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
- `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 __mask`)
- `fmtflags` [setf](#) (`fmtflags __fmtfl`)
- void [setstate](#) (`iosstate __state`)
- `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`

### 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.

- `void _M_write (const char_type *_s, streamsize __n)`
- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *_s, streamsize __n)`

### 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<< (__streambuf_type *_sb)`
- `__ostream_type & operator<< (const void *_p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`

- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`
  
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`

### 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 ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename `_ValueType` >  
`__ostream_type` & `_M_insert` (`_ValueType` \_\_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`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits >> __num_get_type`
- `operator void * () const`
- `bool operator! () const`

### 5.380.1 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 55 of file `ostream`.

### 5.380.2 Member Typedef Documentation

**5.380.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream<_CharT, _Traits >::__ctype_type`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits >`.

Definition at line 71 of file `ostream`.

**5.380.2.2** `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 86 of file `basic_ios.h`.

**5.380.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits>::_num_put_type`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `ostream`.

**5.380.2.4** `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 59 of file `ostream`.

**5.380.2.5** `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 444 of file `ios_base.h`.

**5.380.2.6** `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 263 of file `ios_base.h`.

#### **5.380.2.7 `template<typename _CharT, typename _Traits> typedef` `_Traits::int_type std::basic_ostream< _CharT, _Traits >::int_type`**

These are non-standard types.

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 60 of file `ostream`.

**5.380.2.8** `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 338 of file `ios_base.h`.

**5.380.2.9** `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream<_CharT, _Traits>::off_type`

These are non-standard types.

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`.

**5.380.2.10** `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 369 of file `ios_base.h`.

### 5.380.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ostream< _CharT, _Traits >::pos_type`

These are non-standard types.

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`.

### 5.380.2.12 `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 401 of file `ios_base.h`.

### 5.380.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ostream< _CharT, _Traits >::traits_type`

These are non-standard types.

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`.

## 5.380.3 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 427 of file `ios_base.h`.

## 5.380.4 Constructor & Destructor Documentation

**5.380.4.1** `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 82 of file `ostream`.

**5.380.4.2** `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 91 of file `ostream`.

## 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 705 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]`

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 287 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 201 of file basic\_ios.h.

**5.380.5.4** `template<typename _CharT , typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)  
[inline, inherited]`

[Re]sets the error state.

**Parameters:**

*state* The new state flag(s) to [set](#).

## 5.380 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 2075

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**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) [inline, 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 62 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 180 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()`, 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 247 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 212 of file `basic_ios.h`.

Referenced by `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 191 of file basic\_ios.h.

Referenced by `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 380 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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

#### 5.380.5.12 `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 561 of file `ios_base.h`.

#### 5.380.5.13 `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 550 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<<()`, and `std::operator>>()`.

#### 5.380.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inline]`

Synchronizing the stream buffer.

##### Returns:

\*this

## 5.380 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 2079

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 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 `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 694 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()`.

### 5.380.5.16 `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 170 of file `basic_ios.h`.

### 5.380.5.17 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue(const locale & __loc)` [`inline`, `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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.380.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<_CharT,  
_Traits> * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.380.5.19** `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.



## 5.380 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 2081

Definition at line 740 of file `ios_base.h`.

**5.380.5.20** `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 `l10n` notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.380.5.21** `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 111 of file `basic_ios.h`.

**5.380.5.22** `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 115 of file `basic_ios.h`.

**5.380.5.23** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (_streambuf_type * __sb) [inline]`

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 120 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()`.

**5.380.5.24** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (const void * __p) [inline]`

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 225 of file ostream.

**5.380.5.25** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long double __f) [inline]`

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 221 of file ostream.

**5.380.5.26** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (float __f) [inline]`

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 213 of file ostream.

```
5.380.5.27 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f)
[inline]
```

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 209 of file ostream.

**5.380.5.28** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned long long __n) [inline]`

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 204 of file ostream.

**5.380.5.29** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (long long __n) [inline]`

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 200 of file ostream.

**5.380.5.30** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline]`

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 191 of file ostream.

**5.380.5.31** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (int __n) [inline]`

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 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.380.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline]`

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 180 of file ostream.

**5.380.5.33** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (short __n) [inline]`

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 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.380.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (bool __n) [inline]`

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 173 of file ostream.

**5.380.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long __n) [inline]`

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 169 of file ostream.

**5.380.5.36** `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 165 of file ostream.

**5.380.5.37** `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 127 of file ostream.

**5.380.5.38** `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 117 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]`

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 108 of file `ostream`.

**5.380.5.40** `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 629 of file `ios_base.h`.

**5.380.5.41** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.380.5.42** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::put (char_type __c) [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 149 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()`.

**5.380.5.43 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 761 of file ios\_base.h.

**5.380.5.44 template<typename \_CharT, typename \_Traits> basic\_streambuf<\_CharT, \_Traits > \* std::basic\_ios<\_CharT, \_Traits >::rdbuf (basic\_streambuf<\_CharT, \_Traits > \* \_\_sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

#### 5.380.5.45 `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_istringstream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `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 >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.380.5.46** `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 127 of file `basic_ios.h`.

**5.380.5.47** `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.

**5.380.5.48** `template<typename _CharT , typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp  
(off_type __off, ios_base::seekdir __dir) [inline]`

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 290 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()`.

#### 5.380.5.49 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inline]`

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 258 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()`.

#### 5.380.5.50 `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 594 of file `ios_base.h`.

#### 5.380.5.51 `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

#### 5.380.5.52 `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 147 of file `basic_ios.h`.

Referenced by `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_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<`



## 5.380 std::basic\_ostream< \_CharT, \_Traits > Class Template Reference 2097

`_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 >::unget()`, and `std::ws()`.

### 5.380.553 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>

### 5.380.554 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits >::pos\_type std::basic\_ostream< \_CharT, \_Traits >::tellp () [inline]

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 237 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()`.

### 5.380.555 template<typename \_CharT, typename \_Traits> basic\_ostream< \_CharT, \_Traits >\* std::basic\_ios< \_CharT, \_Traits >::tie (basic\_ostream< \_CharT, \_Traits > \* \_\_tistr) [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 297 of file `basic_ios.h`.

```
5.380.5.56 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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

```
5.380.5.57 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 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.380.5.58** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.380.5.59** `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 652 of file `ios_base.h`.

**5.380.5.60** `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 643 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.380.5.61** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::write(const char_type * __s, streamsize __n) [inline]`

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 183 of file ostream.tcc.

References `std::basic_ostream<_CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.380.5.62** `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.

## **5.380.6 Member Data Documentation**

### **5.380.6.1 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 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

### **5.380.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

### **5.380.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### **5.380.6.4 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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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 >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_ostream< _CharT, _Traits >::write()`.

#### 5.380.6.5 `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 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.380.6.6 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### 5.380.6.7 `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 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### 5.380.6.8 `const fmtflags std::ios_base::boolalpha` [`static`, `inherited`]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.380.6.9 `const seekdir std::ios_base::cur` [`static`, `inherited`]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

## 5.380 std::basic\_ostream< \_CharT, \_Traits > Class Template Reference 2103

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), 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().

### 5.380.6.10 const fmtflags std::ios\_base::dec [static, inherited]

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file ios\_base.h.

### 5.380.6.11 const seekdir std::ios\_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

### 5.380.6.12 const iostate std::ios\_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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\_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 >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

### 5.380.6.13 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 350 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\_istream< \_

CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), 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(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

#### 5.380.6.14 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

#### 5.380.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios\_base.h.

#### 5.380.6.16 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 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 >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 5.380.6.17 const fmtflags std::ios\_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().



**5.380.6.18 `const openmode std::ios_base::in` [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `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_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()`.

**5.380.6.19 `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 280 of file `ios_base.h`.

**5.380.6.20 `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter>::do_put()`.

**5.380.6.21 `const fmtflags std::ios_base::oct` [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.380.6.22 `const openmode std::ios_base::out` [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `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()`.

**5.380.6.23 `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 291 of file `ios_base.h`.

**5.380.6.24 `const fmtflags std::ios_base::scientific` [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.380.6.25 `const fmtflags std::ios_base::showbase` [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file `ios_base.h`.

**5.380.6.26 `const fmtflags std::ios_base::showpoint` [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file `ios_base.h`.

**5.380.6.27 `const fmtflags std::ios_base::showpos` [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

**5.380.6.28 `const fmtflags std::ios_base::skipws` [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

**5.380.6.29 `const openmode std::ios_base::trunc` [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

## **5.380 std::basic\_ostream<\_CharT, \_Traits > Class Template Reference 2107**

### **5.380.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

### **5.380.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

## 5.381 `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

### 5.381.1 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 377 of file ostream.

### 5.381.2 Constructor & Destructor Documentation

**5.381.2.1** `template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::sentry::sentry (basic_ostream< _CharT, _Traits > & __os) [inline, 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 47 of file ostream.tcc.

**5.381.2.2** `template<typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits>::sentry::~sentry ()`  
`[inline]`

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 405 of file `ostream`.

References `std::uncaught_exception()`, and `std::unitbuf()`.

### 5.381.3 Member Function Documentation

**5.381.3.1** `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 426 of file `ostream`.

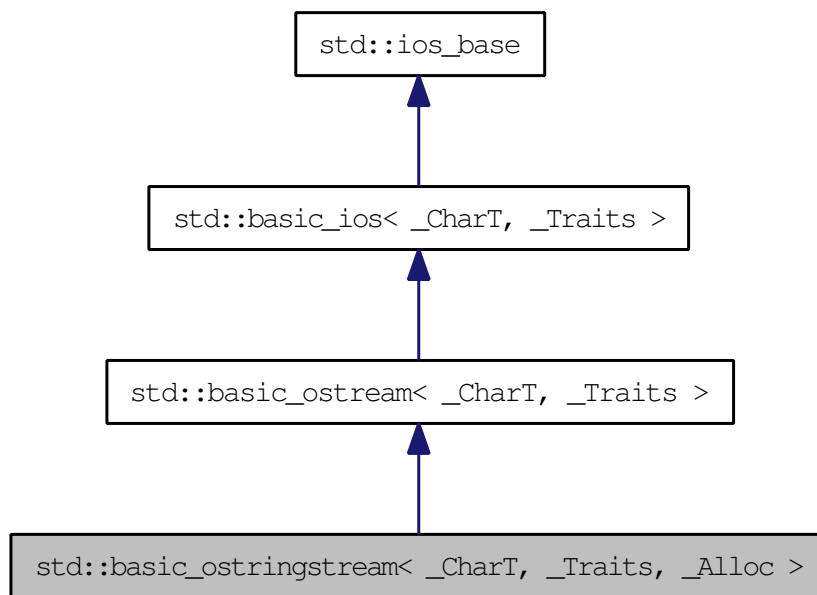
The documentation for this class was generated from the following files:

- `ostream`
- `ostream.tcc`

## 5.382 `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](#)

## Public Member Functions

- [basic\\_ostringstream](#) (const `__string_type` &\_\_str, ios\_base::openmode \_\_mode=ios\_base::out)
- [basic\\_ostringstream](#) (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
- void [exceptions](#) (iostate \_\_except)
- `iostate` [exceptions](#) () const
- bool [fail](#) () const
- `char_type` [fill](#) (`char_type` \_\_ch)
- `char_type` [fill](#) () const
- `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](#) (`streamsize` \_\_prec)
- `streamsize` [precision](#) () const

- `void *& pword (int __ix)`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `__stringbuf_type * rdbuf () const`
- `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 __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `void setstate (iosstate __state)`
- `void str (const __string_type & __s)`
- `__string_type str () const`
- `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`

### 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.

- `void _M_write (const char_type * __s, streamsize __n)`
- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type * __s, streamsize __n)`

### 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<< (__streambuf_type * __sb)`
- `__ostream_type & operator<< (const void * __p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`
  
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`

### 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`
- `typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> __num_get_type`
- `operator void * () const`
- `bool operator! () const`

### 5.382.1 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 366 of file `sstream`.

## 5.382.2 Member Typedef Documentation

**5.382.2.1** `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 71 of file ostream.

**5.382.2.2** `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 86 of file basic\_ios.h.

**5.382.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream< _CharT, _Traits >::_num_put_type  
[inherited]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.382.2.4** `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 370 of file sstream.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2117

---

### 5.382.2.5 `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 444 of file `ios_base.h`.

### 5.382.2.6 `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 263 of file ios\_base.h.

**5.382.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::int_type std::basic_ostringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 375 of file sstream.

**5.382.2.8** `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 338 of file ios\_base.h.

**5.382.2.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::off_type std::basic_ostringstream< _CharT,  
_Traits, _Alloc >::off_type`

These are non-standard types.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 377 of file sstream.

**5.382 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference 2119**

---

**5.382.2.10 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 369 of file `ios_base.h`.

**5.382.2.11 template<typename \_CharT, typename \_Traits, typename \_Alloc> typedef traits\_type::pos\_type std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc >::pos\_type**

These are non-standard types.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Definition at line 376 of file `sstream`.

**5.382.2.12 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 401 of file `ios_base.h`.

**5.382.2.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
 typedef _Traits std::basic_ostringstream< _CharT, _Traits, _Alloc  
 >::traits_type`

These are non-standard types.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Definition at line 371 of file `sstream`.

### 5.382.3 Member Enumeration Documentation

**5.382.3.1** `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 427 of file `ios_base.h`.

### 5.382.4 Constructor & Destructor Documentation

**5.382.4.1** `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 402 of file `sstream`.



**5.382.4.2** `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 420 of file `sstream`.

**5.382.4.3** `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 431 of file `sstream`.

## 5.382.5 Member Function Documentation

**5.382.5.1** `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 705 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()`.

**5.382.5.2** `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 287 of file ostream.

Referenced by `std::basic_ostream< _CharT, _Traits >::write()`.

**5.382.5.3** `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 201 of file basic\_ios.h.

**5.382.5.4** `template<typename _CharT, typename _Traits > void std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit) [inline, inherited]`

[Re]sets the error state.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2123

### 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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.382.5.5** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt(const basic_ios<_CharT, _Traits> & __rhs) [inline, 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 62 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()`.

**5.382.5.6** `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 180 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()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.382.5.7 `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 247 of file `basic_ios.h`.

#### 5.382.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline, inherited]`

Throwing exceptions on errors.

##### Returns:

The current exceptions mask.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2125

---

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.382.5.9 `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 191 of file `basic_ios.h`.

Referenced by `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()`.

### 5.382.5.10 `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 380 of file `basic_ios.h`.

**5.382.5.11** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.382.5.12** `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 561 of file `ios_base.h`.

**5.382.5.13** `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 550 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<<()`, and `std::operator>>()`.

**5.382 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 2127

---

**5.382.5.14** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inline, 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 211 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()`.

**5.382.5.15** `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 694 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()`.

**5.382.5.16** `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 170 of file `basic_ios.h`.

**5.382.5.17** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc)  
[inline, 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/localizati>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.382.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<_CharT,  
_Traits> * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.382.5.19** `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.



## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2129

---

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 740 of file `ios_base.h`.

**5.382.5.20** `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 notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.382.5.21** `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 111 of file `basic_ios.h`.

**5.382.5.22** `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 115 of file `basic_ios.h`.

**5.382.5.23** `template<typename _CharT, typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits  
>::operator<< (_streambuf_type * __sb) [inline,  
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 120 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()`.

**5.382.5.24** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< (const void *  
__p) [inline, 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 225 of file ostream.

**5.382.5.25** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long double __f) [inline, 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 221 of file ostream.

```
5.382.5.26 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, 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 213 of file ostream.

```
5.382.5.27 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f)
[inline, 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.

**5.382 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2133

---

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 209 of file ostream.

**5.382.5.28** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long long __n) [inline, 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 204 of file ostream.

**5.382.5.29** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline, 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 200 of file ostream.

**5.382.5.30** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline, 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,

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2135

- 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 191 of file ostream.

**5.382.5.31** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n) [inline, 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 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.382.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned short __n) [inline, 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 180 of file ostream.

```
5.382.5.33 template<typename _CharT, typename _Traits > basic_ostream<
 _CharT, _Traits > & std::basic_ostream< _CharT, _Traits
 >::operator<< (short __n) [inline, 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



## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2137

---

If the function inserts no characters, failbit is [set](#).

Definition at line 92 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

### 5.382.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (bool __n) [inline, 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 173 of file `ostream`.

### 5.382.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long __n) [inline, 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 169 of file ostream.

**5.382.5.36** `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 165 of file ostream.

**5.382.5.37** `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 127 of file ostream.

**5.382 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 2139

---

**5.382.5.38** `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 117 of file `ostream`.

**5.382.5.39** `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 `omanip` header.

Definition at line 108 of file `ostream`.

**5.382.5.40** `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 629 of file `ios_base.h`.

**5.382.5.41** `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 620 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.382.5.42** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put(char_type __c) [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 149 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()`.

**5.382.5.43** `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.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2141

Definition at line 761 of file `ios_base.h`.

**5.382.5.44** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf(basic_streambuf<_CharT, _Traits> * __sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.382.5.45** `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 442 of file `sstream`.

**5.382.5.46** `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 127 of file `basic_ios.h`.

**5.382.5.47** `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.

**5.382.5.48** `template<typename _CharT , typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp  
(off_type __off, ios_base::seekdir __dir) [inline, 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.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2143

---

Definition at line 290 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()`.

### 5.382.5.49 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos)` [`inline`, `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 258 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()`.

### 5.382.5.50 `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 594 of file `ios_base.h`.

**5.382.5.51** `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.382.5.52** `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 147 of file `basic_ios.h`.

Referenced by `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_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 >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.



**5.382 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **2145**

---

**5.382.5.53** `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 460 of file `sstream`.

**5.382.5.54** `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 450 of file `sstream`.

Referenced by `std::operator<<()`.

**5.382.5.55** `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>

**5.382.5.56** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inline, 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 237 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()`.

**5.382.5.57** `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 297 of file `basic_ios.h`.

**5.382.5.58** `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.

## 5.382 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2147

---

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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.382.5.59 `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 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

### 5.382.5.60 `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

```
std::use_facet<ctype<char_type>> (getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.382.5.61** `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 652 of file `ios_base.h`.

**5.382.5.62** `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 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator<>>()`.

**5.382.5.63** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n) [inline, 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:

## 5.382 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 2149

---

- $n$  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 183 of file `ostream.tcc`.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

### 5.382.5.64 `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.

## 5.382.6 Member Data Documentation

### 5.382.6.1 `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 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

### 5.382.6.2 `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.382.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.382.6.4 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 342 of file ios\_base.h.

Referenced by 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::operator<<(), 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 >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::basic\_ostream< \_CharT, \_Traits >::write().

**5.382.6.5 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 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.382.6.6 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

## 5.382 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 2151

---

### 5.382.6.7 `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 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

### 5.382.6.8 `const fmtflags std::ios_base::boolalpha` [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

### 5.382.6.9 `const seekdir std::ios_base::cur` [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `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()`.

### 5.382.6.10 `const fmtflags std::ios_base::dec` [static, inherited]

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file `ios_base.h`.

### 5.382.6.11 `const seekdir std::ios_base::end` [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**5.382.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::ws()`.

**5.382.6.13 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 350 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_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `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()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

**5.382.6.14 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.382.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios\_base.h.



**5.382.6.16 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.382.6.17 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.382.6.18 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios\_base.h.

Referenced by 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\_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().

**5.382.6.19 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 280 of file `ios_base.h`.

#### **5.382.6.20 `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

#### **5.382.6.21 `const fmtflags std::ios_base::oct` [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### **5.382.6.22 `const openmode std::ios_base::out` [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf<_CharT, _Traits >::overflow()`, `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()`.

#### **5.382.6.23 `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 291 of file `ios_base.h`.

#### **5.382.6.24 `const fmtflags std::ios_base::scientific` [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.382.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file ios\_base.h.

**5.382.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file ios\_base.h.

**5.382.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.382.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.382.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.382.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.382.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

- [sstream](#)

## 5.383 `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` (`initializer_list< _Ch_type > __l`, `flag_type __f=regex_constants::ECMAScript`)
- `template<typename _InputIterator > basic_regex` (`_InputIterator __first`, `_InputIterator __last`, `flag_type __f=regex_constants::ECMAScript`)
- `template<typename _Ch_traits, typename _Ch_alloc > basic_regex` (`const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s`, `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`)
- `basic_regex` (`const _Ch_type *__p`, `flag_type __f=regex_constants::ECMAScript`)
- `basic_regex` ()
- `~basic_regex` ()
- `basic_regex & assign` (`initializer_list< _Ch_type > __l`, `flag_type __f=regex_constants::ECMAScript`)
- `template<typename _InputIterator > basic_regex & assign` (`_InputIterator __first`, `_InputIterator __last`, `flag_type __f=regex_constants::ECMAScript`)
- `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` (`const _Ch_type *__p`, `std::size_t __len`, `flag_type __f`)
- `basic_regex & assign` (`const _Ch_type *__p`, `flag_type __f=regex_constants::ECMAScript`)
- `basic_regex & assign` (`const basic_regex &__that`)
- `flag_type flags` () const
- `locale_type getloc` () const
- `locale_type imbue` (`locale_type __loc`)
- `unsigned int mark_count` () const

- `template<typename _Ch_traits, typename _Allocator >`  
`basic_regex & operator= (const basic_string< _Ch_type, _Ch_traits, _`  
`Allocator > &__s)`
- `basic_regex & operator= (const _Ch_type *__p)`
- `basic_regex & operator= (const basic_regex &__rhs)`
- `void swap (basic_regex &__rhs)`

## Static Public Attributes

### Constants

*tr1* [7.8.1] *std* [28.8.1]

- static const `regex_constants::syntax_option_type` **awk**
- static const `regex_constants::syntax_option_type` **basic**
- static const `regex_constants::syntax_option_type` **collate**
- static const `regex_constants::syntax_option_type` **ECMAScript**
- static const `regex_constants::syntax_option_type` **egrep**
- static const `regex_constants::syntax_option_type` **extended**
- static const `regex_constants::syntax_option_type` **grep**
- static const `regex_constants::syntax_option_type` **icase**
- static const `regex_constants::syntax_option_type` **nosubs**
- static const `regex_constants::syntax_option_type` **optimize**

## Protected Attributes

- `flag_type` **\_M\_flags**
- `unsigned int` **\_M\_mark\_count**
- `string_type` **\_M\_pattern**
- `_Rx_traits` **\_M\_traits**

### 5.383.1 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`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 711 of file `tr1_impl/regex`.

## 5.383.2 Constructor & Destructor Documentation

**5.383.2.1** `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 752 of file tr1\_impl/regex.

**5.383.2.2** `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 768 of file tr1\_impl/regex.

**5.383.2.3** `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 784 of file `tr1_impl/regex`.

```
5.383.2.4 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 793 of file `tr1_impl/regex`.

```
5.383.2.5 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 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 809 of file `tr1_impl/regex`.



**5.383.2.6** `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 828 of file tr1\_impl/regex.

**5.383.2.7** `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 842 of file tr1\_impl/regex.

**5.383.2.8** `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 851 of file tr1\_impl/regex.

### 5.383.3 Member Function Documentation

**5.383.3.1** `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.  
*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 984 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

**5.383.3.2** `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 968 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

## 5.383 std::basic\_regex<\_Ch\_type, \_Rx\_traits > Class Template Reference 2163

**5.383.3.3** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _Ch_type, traits, _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.

### 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 945 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

**5.383.3.4** `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 929 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

**5.383.3.5** `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 911 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

**5.383.3.6** `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 > & __that) [inline]`

the real assignment operator.

**Parameters:**

- that* Another regular expression object.

Definition at line 890 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::operator=()`.

**5.383.3.7** `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()`.

## 5.383 std::basic\_regex<\_Ch\_type, \_Rx\_traits > Class Template Reference 2165

Definition at line 1003 of file tr1\_impl/regex.

Referenced by std::basic\_regex<\_Ch\_type, \_Rx\_traits >::operator=().

**5.383.3.8** `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 1021 of file tr1\_impl/regex.

**5.383.3.9** `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 1013 of file tr1\_impl/regex.

**5.383.3.10** `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 995 of file tr1\_impl/regex.

**5.383.3.11** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _Ch_typeraits  
, typename _Allocator > basic_regex& std::basic_regex<  
_Ch_type, _Rx_traits >::operator= (const basic_string<_Ch_type,  
_Ch_typeraits, _Allocator > & __s) [inline]`

Replaces a regular expression with a new one constructed from a string.

### Parameters:

*A* a pointer to a string containing a regular expression.

Definition at line 880 of file tr1\_impl/regex.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits >::flags()`.

**5.383.3.12** `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 869 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits >::flags()`.

**5.383.3.13** `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 858 of file `tr1_impl/regex`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

**5.383.3.14** `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 1031 of file `tr1_impl/regex`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

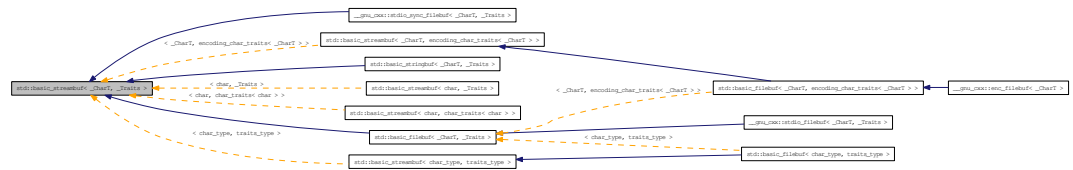
**5.383 std::basic\_regex<\_Ch\_type, \_Rx\_traits > Class Template Reference 2167**

- [tr1\\_impl/regex](#)

## 5.384 `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 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 \(const char\\_type \\* \\_\\_s, streamsize \\_\\_n\)](#)
- [void stoss \(\)](#)
- [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 xsputn` (`const char_type * __s`, `streamsize __n`)

## 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 >`
- 
- `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`

- `locale _M_buf_locale`
- `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`
- `virtual ~basic_streambuf ()`
- `locale getloc () const`
- `locale pubimbue (const locale &__loc)`
- `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)`
- `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
- `int pubsync ()`
- `char_type * eback () const`
- `char_type * egptr () const`
- `char_type * epptr () const`
- `char_type * gptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`

### 5.384.1 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 115 of file `streambuf`.

## 5.384.2 Member Typedef Documentation

**5.384.2.1** `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 133 of file streambuf.

**5.384.2.2** `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 124 of file streambuf.

**5.384.2.3** `template<typename _CharT, typename _Traits> typedef  
traits_type::int_type std::basic_streambuf< _CharT, _Traits  
>::int_type`

This is a non-standard type.

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.

**5.384.2.4** `template<typename _CharT, typename _Traits> typedef  
traits_type::off_type std::basic_streambuf< _CharT, _Traits  
>::off_type`

This is a non-standard type.

## 5.384 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2173

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.

**5.384.2.5** `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_streambuf<_CharT, _Traits>::pos_type`

This is a non-standard type.

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.

**5.384.2.6** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_streambuf<_CharT, _Traits>::traits_type`

This is a non-standard type.

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 125 of file streambuf.

## 5.384.3 Constructor & Destructor Documentation

**5.384.3.1** `template<typename _CharT, typename _Traits> virtual std::basic_streambuf<_CharT, _Traits>::~basic_streambuf () [inline, virtual]`

Destructor deallocates no buffer space.

Definition at line 193 of file streambuf.

**5.384.3.2** `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 441 of file streambuf.

## 5.384.4 Member Function Documentation

**5.384.4.1** `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 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.384.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr () const [inline,  
protected]`

Locale access.

### Returns:

The current [locale](#) in effect.

## 5.384 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2175

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 466 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

### 5.384.4.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eptr() const` [`inline`, `protected`]

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 513 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### 5.384.4.4 `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 476 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.384.4.5** `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 222 of file `streambuf`.

**5.384.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::gptr () const [inline,  
protected]`

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 463 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.384.4.7** `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](#).



## 5.384 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2177

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 554 of file streambuf.

**5.384.4.8** `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 262 of file streambuf.

**5.384.4.9** `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 746 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits >::xsputn()`.

**5.384.4.10** `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

**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 702 of file streambuf.

**5.384.4.11** `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 507 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_filebuf< _CharT, _Traits >::sync()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.384.4.12** `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 523 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()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.384.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr () const  
[inline, protected]`

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 510 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_filebuf<_CharT, _Traits>::sync()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.384.4.14** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::pubimbue (const locale  
& __loc) [inline]`

Entry point for `imbue()`.

**Parameters:**

*loc* The new [locale](#).

**Returns:**

The previous [locale](#).

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

**5.384.4.15** `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]`

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 239 of file `streambuf`.

**5.384.4.16** `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]`

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 244 of file `streambuf`.

**5.384.4.17** `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 235 of file `streambuf`.

**5.384.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync () [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 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.384.4.19** `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 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

**5.384.4.20** `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 580 of file `streambuf`.

**5.384.4.21** `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.

**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 592 of file `streambuf`.

**5.384.4.22** `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 569 of file `streambuf`.

**5.384.4.23** `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 487 of file `streambuf`.

**5.384.4.24** `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* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.384.4.25** `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.

Definition at line 316 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()`.

**5.384.4.26** `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 335 of file `streambuf`.



**5.384.4.27** `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.*

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 627 of file `streambuf`.

**5.384.4.28** `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 `sputc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 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()`.

**5.384.4.29** `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*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.384.4.30** `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 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

## 5.384 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 2187

**5.384.4.31** `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 `xspun(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.384.4.32** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stossc () [inline]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.384.4.33** `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 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.384.4.34** `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 605 of file streambuf.

**5.384.4.35** `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 678 of file streambuf.

**5.384.4.36** `template<typename _CharT, typename _Traits> virtual  
int_type std::basic_streambuf< _CharT, _Traits >::underflow ()  
[inline, protected, virtual]`

Fetches more data from the controlled sequence.

## 5.384 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2189

### 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 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 665 of file `streambuf`.

```
5.384.4.37 template<typename _CharT , typename _Traits > streamsize
std::basic_streambuf< _CharT, _Traits >::xsggetn (char_type * __s,
streamsize __n) [inline, 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 45 of file `streambuf.tcc`.

References `std::min()`.

**5.384.4.38** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsputn(const char_type  
* __s, streamsize __n) [inline, 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 79 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()`.

## 5.384.5 Member Data Documentation

**5.384.5.1** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale  
[protected]`

Current `locale` setting.

Definition at line 188 of file `streambuf`.

## 5.384 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 2191

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::basic\_filebuf().

### **5.384.5.2** `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 180 of file `streambuf`.

### **5.384.5.3** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected]`

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 181 of file `streambuf`.

### **5.384.5.4** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected]`

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 182 of file `streambuf`.

**5.384.5.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected]`

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 183 of file `streambuf`.

**5.384.5.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected]`

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 184 of file `streambuf`.

**5.384.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected]`

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 185 of file `streambuf`.

The documentation for this class was generated from the following files:

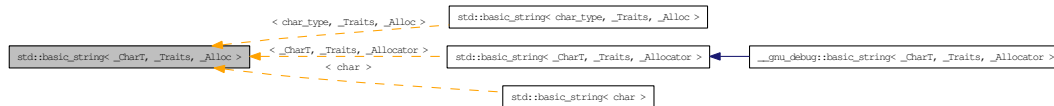
- [streambuf](#)
- [streambuf.tcc](#)



## 5.385 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference 2193

### 5.385 `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

- `template<class InputIterator > basic_string (InputIterator __beg, InputIterator __end, const _Alloc &__a= _Alloc())`
- `basic_string (initializer_list<_CharT > __l, const _Alloc &__a=_Alloc())`
- `basic_string (basic_string &&__str)`
- `basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())`
- `basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `basic_string (const basic_string &__str, size_type __pos, size_type __n=npos)`

- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [\\_Alloc](#) &\_\_a)
- [basic\\_string](#) ()
- [~basic\\_string](#) ()
- [template<typename \\_InIterator > \\_CharT \\* \*\*\\_S\\_construct\*\* \(\\_InIterator \\_\\_beg, \\_InIterator \\_\\_end, const \[\\\_Alloc\]\(#\) &\\_\\_a, \[forward\\\_iterator\\\_tag\]\(#\)\)](#)
- [template<class \\_InputIterator > \[basic\\\_string\]\(#\) & \[append\]\(#\) \(\\_InputIterator \\_\\_first, \\_InputIterator \\_\\_last\)](#)
- [basic\\_string](#) & [append](#) ([initializer\\_list](#)< [\\_CharT](#) > \_\_l)
- [basic\\_string](#) & [append](#) (size\_type \_\_n, [\\_CharT](#) \_\_c)
- [basic\\_string](#) & [append](#) (const [\\_CharT](#) \*\_\_s)
- [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 [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) ([initializer\\_list](#)< [\\_CharT](#) > \_\_l)
- [template<class \\_InputIterator > \[basic\\\_string\]\(#\) & \[assign\]\(#\) \(\\_InputIterator \\_\\_first, \\_InputIterator \\_\\_last\)](#)
- [basic\\_string](#) & [assign](#) (size\_type \_\_n, [\\_CharT](#) \_\_c)
- [basic\\_string](#) & [assign](#) (const [\\_CharT](#) \*\_\_s)
- [basic\\_string](#) & [assign](#) (const [\\_CharT](#) \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [reference at](#) (size\_type \_\_n)
- [const\\_reference at](#) (size\_type \_\_n) const
- [const\\_iterator begin](#) () const
- [iterator begin](#) ()
- [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, size\_type \_\_n2) const
- [int compare](#) (size\_type \_\_pos, size\_type \_\_n1, const [\\_CharT](#) \*\_\_s) const
- [int compare](#) (const [\\_CharT](#) \*\_\_s) 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 \_\_n, const [basic\\_string](#) &\_\_str) const
- [int compare](#) (const [basic\\_string](#) &\_\_str) const

## 5.385 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference

- `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`
- `const_iterator end` () `const`
- `iterator end` ()
- `iterator erase` (`iterator __first`, `iterator __last`)
- `iterator erase` (`iterator __position`)
- `basic_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `size_type find` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find` (`const _CharT *__s`, `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_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`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` (`const basic_string &__str`, `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=0`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `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=npo`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=npo`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_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`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=npo`) `const`
- `allocator_type get_allocator` () `const`
- `iterator insert` (`iterator __p`, `_CharT __c`)
- `basic_string & insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `basic_string & insert` (`size_type __pos`, `const _CharT *__s`)
- `basic_string & insert` (`size_type __pos`, `const _CharT *__s`, `size_type __n`)
- `basic_string & insert` (`size_type __pos1`, `const basic_string &__str`, `size_type __pos2`, `size_type __n`)
- `basic_string & insert` (`size_type __pos1`, `const basic_string &__str`)

- void [insert](#) (iterator \_\_p, [initializer\\_list](#)< \_CharT > \_\_l)
- [template](#)<class \_InputIterator >  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- size\_type [length](#) () const
- size\_type [max\\_size](#) () const
- [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- reference [operator\[\]](#) (size\_type \_\_pos)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const
- void [push\\_back](#) (\_CharT \_\_c)
- [const\\_reverse\\_iterator](#) [rbegin](#) () const
- [reverse\\_iterator](#) [rbegin](#) ()
- [const\\_reverse\\_iterator](#) [rend](#) () const
- [reverse\\_iterator](#) [rend](#) ()
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [initializer\\_list](#)< \_CharT > \_\_l)
- [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, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- [template](#)<class \_InputIterator >  
[basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [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)

## 5.385 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference

- `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 \_\_n, const `basic_string` &\_\_str)
- 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=`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
- 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`

### 5.385.1 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

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Documentation? What's that? Nathan Myers <[ncm@cantrip.org](mailto:ncm@cantrip.org)>.

A string looks like this:

```
[basic_string<char_type>]
_M_dataplus
_M_p ----->
 [_Rep]
 _M_length
 _M_capacity
 _M_refcount
 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 104 of file `basic_string.h`.

## 5.385.2 Constructor & Destructor Documentation

**5.385.2.1** `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 422 of file `basic_string.h`.

**5.385.2.2** `template<typename _CharT , typename _Traits , typename _Alloc>  
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const  
_Alloc & __a) [inline, explicit]`

Construct an empty string using [allocator](#) *a*.

Reimplemented in `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 178 of file `basic_string.tcc`.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2199

**5.385.2.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (const  
basic_string<_CharT, _Traits, _Alloc > & __str) [inline]`

Construct string with copy of value of *str*.

### Parameters:

*str* Source string.

Definition at line 170 of file basic\_string.tcc.

**5.385.2.4** `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) [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 184 of file basic\_string.tcc.

**5.385.2.5** `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) [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 194 of file basic\_string.tcc.

**5.385.2.6** `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 ())  
[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 206 of file `basic_string.tcc`.

**5.385.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string< _CharT, _Traits, _Alloc >::basic_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](#)).

Reimplemented in `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 213 of file `basic_string.tcc`.

**5.385.2.8** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (size_type  
__n, _CharT __c, const _Alloc & __a = _Alloc ()) [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 220 of file `basic_string.tcc`.



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 201

**5.385.2.9** `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 492 of file basic\_string.h.

**5.385.2.10** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string  
(initializer_list<_CharT> __l, const _Alloc & __a = _Alloc())  
[inline]`

Construct string from an initializer [list](#).

### Parameters:

*l* `std::initializer_list` of characters.

*a* Allocator to use (default is default [allocator](#)).

Reimplemented in `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 235 of file basic\_string.tcc.

**5.385.2.11** `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())  
[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](#)).

Reimplemented in `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`.

Definition at line 228 of file basic\_string.tcc.

**5.385.2.12** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ()  
[inline]`

Destroy the string instance.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 523 of file `basic_string.h`.

### 5.385.3 Member Function Documentation

**5.385.3.1** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 991 of file `basic_string.h`.

**5.385.3.2** `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.

### 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

Definition at line 977 of file basic\_string.h.

Referenced by std::basic\_string<char >::append().

```
5.385.3.3 template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::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 281 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().

```
5.385.3.4 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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 953 of file basic\_string.h.

```
5.385.3.5 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) [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 298 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()`.

```
5.385.3.6 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)
 [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.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

Definition at line 342 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()`.

**5.385.3.7** `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) [inline]`

Append a string to this string.

### Parameters:

*str* The string to append.

### Returns:

Reference to this string.

Definition at line 325 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::collate<_CharT >::do_transform()`, `std::operator+()`, `std::operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc >::resize()`.

**5.385.3.8** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign  
(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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 1113 of file basic\_string.h.

Referenced by `std::basic_string<char >::assign()`.

**5.385.3.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
 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).

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 1103 of file basic\_string.h.

**5.385.3.10** `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 1090 of file basic\_string.h.

**5.385.3.11** `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.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

### 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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 1074 of file `basic_string.h`.

```
5.385.3.12 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) [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 259 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

```
5.385.3.13 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 1046 of file `basic_string.h`.

Referenced by `std::basic_string< char >::assign()`.

**5.385.3.14** `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 1026 of file `basic_string.h`.

**5.385.3.15** `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) [inline]`

Set value to contents of another string.

**Parameters:**

*str* Source string to use.



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2209

### Returns:

Reference to this string.

Definition at line 243 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::get\_allocator().

Referenced by std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc >::overflow(), and std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc >::str().

**5.385.3.16** `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 865 of file basic\_string.h.

**5.385.3.17** `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 846 of file `basic_string.h`.

**5.385.3.18** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 601 of file `basic_string.h`.

**5.385.3.19** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 590 of file `basic_string.h`.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

**5.385.3.20** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 1722 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()`.

## 5.385 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 2211

**5.385.3.21** `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 758 of file basic\_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

**5.385.3.22** `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 665 of file basic\_string.h.

**5.385.3.23** `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 673 of file basic\_string.h.

**5.385.3.24** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string< _CharT, _Traits, _Alloc >::clear ()  
[inline]`

Erases the string, making it empty.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 785 of file basic\_string.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf()`.

**5.385.3.25** `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 [inline]`

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 *r1en* 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,r1en)`. 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 980 of file `basic_string.tcc`.

References `std::min()`.

**5.385.3.26** `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  
[inline]`

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.

## 5.385 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference 2213

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 964 of file `basic_string.tcc`.

References `std::min()`.

**5.385.3.27** `template<typename _CharT, typename _Traits, typename _Alloc > int std::basic_string<_CharT, _Traits, _Alloc >::compare (const _CharT * __s) const [inline]`

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.

Reimplemented in `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >`.

Definition at line 949 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()`.

**5.385.3.28** `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 [inline]`

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 931 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, and `std::min()`.

**5.385.3.29** `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 [inline]`

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

## 5.385 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference

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 916 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()`.

**5.385.3.30** `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 2129 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

**5.385.3.31** `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 [inline]`

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 723 of file `basic_string.tcc`.

**5.385.3.32** `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 682 of file `basic_string.h`.

**5.385.3.33** `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 691 of file `basic_string.h`.

**5.385.3.34** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 1732 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`



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference

>::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< \_Ch\_type >::transform().

**5.385.3.35** `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 793 of file `basic_string.h`.

Referenced by `std::operator>>()`.

**5.385.3.36** `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.

Reimplemented in `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 620 of file `basic_string.h`.

**5.385.3.37** `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.

Reimplemented in `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 609 of file `basic_string.h`.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

**5.385.3.38** `template<typename _CharT, typename _Traits, typename  
_Alloc > basic_string< _CharT, _Traits, _Alloc >::iterator  
std::basic_string< _CharT, _Traits, _Alloc >::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 391 of file basic\_string.tcc.

**5.385.3.39** `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 1319 of file basic\_string.h.

**5.385.3.40** `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.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

### 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 1303 of file `basic_string.h`.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc >::resize()`.

**5.385.3.41** `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 [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 *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 760 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::find()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**5.385.3.42** `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 1781 of file `basic_string.h`.

```
5.385.3.43 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 1767 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find()`.

```
5.385.3.44 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 [inline]
```

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.

### 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2221

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 737 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::size().

Referenced by std::basic\_string<\_CharT, \_Traits, \_Alloc >::find(), and std::basic\_string<\_CharT, \_Traits, \_Alloc >::find\_first\_of().

```
5.385.3.45 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 [inline]
```

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 864 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::size().

```
5.385.3.46 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 2020 of file `basic_string.h`.

```
5.385.3.47 template<typename _CharT, typename _Traits, typename
 _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
 std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of
 (const _CharT * __s, size_type __pos, size_type __n) const
 [inline]
```

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 852 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.385.3.48 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.

### 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

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 1991 of file `basic_string.h`.

Referenced by `std::basic_string<char >::find_first_not_of()`.

**5.385.3.49** `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 1916 of file `basic_string.h`.

**5.385.3.50** `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).

#### **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 1897 of file `basic_string.h`.

**5.385.3.51** `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 [inline]`

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 816 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::find()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.385.3.52** `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.

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 1869 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_first_of()`.



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference

**5.385.3.53** `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 [inline]`

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 896 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.385.3.54** `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 2079 of file `basic_string.h`.

**5.385.3.55** `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 [inline]`

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 875 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.385.3.56** `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 = npow) 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 2050 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2227

**5.385.3.57** `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.
- 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 1977 of file `basic_string.h`.

**5.385.3.58** `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 1958 of file `basic_string.h`.

**5.385.3.59** `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 [inline]`

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 831 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.385.3.60** `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 1930 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_last_of()`.

**5.385.3.61** `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 1739 of file `basic_string.h`.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

Referenced by `std::basic_string<_CharT, _Traits, _Alloc >::assign()`, `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc >::swap()`.

**5.385.3.62** `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 1279 of file `basic_string.h`.

**5.385.3.63** `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 1262 of file `basic_string.h`.

**5.385.3.64** `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 1239 of file `basic_string.h`.

**5.385.3.65** `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) [inline]`

Insert a C substring.

**Parameters:**

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

### 5.385 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2231

*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 360 of file `basic_string.tcc`.

```
5.385.3.66 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 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 1198 of file `basic_string.h`.

Referenced by `std::basic_string<char>::insert()`.

```
5.385.3.67 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 1176 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

```
5.385.3.68 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 1157 of file `basic_string.h`.



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

**5.385.3.69** `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 1146 of file `basic_string.h`.

**5.385.3.70** `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 1130 of file `basic_string.h`.

**5.385.3.71** `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 706 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, `std::match_results< _Bi_iter >::length()`, and `std::regex_traits< _Ch_type >::length()`.

**5.385.3.72** `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 711 of file `basic_string.h`.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`.

**5.385.3.73** `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.

Reimplemented in `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`.

Definition at line 911 of file `basic_string.h`.

**5.385.3.74** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator+= (_CharT __c) [inline]`

Append a character.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

### Parameters:

*c* The character to append.

### Returns:

Reference to this string.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 898 of file `basic_string.h`.

```
5.385.3.75 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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 889 of file `basic_string.h`.

```
5.385.3.76 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 880 of file `basic_string.h`.

**5.385.3.77** `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](#).

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 577 of file `basic_string.h`.

**5.385.3.78** `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.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 565 of file `basic_string.h`.

**5.385.3.79** `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`.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 550 of file `basic_string.h`.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference2237

**5.385.3.80** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 539 of file `basic_string.h`.

**5.385.3.81** `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 531 of file `basic_string.h`.

**5.385.3.82** `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 825 of file `basic_string.h`.

```
5.385.3.83 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 808 of file `basic_string.h`.

```
5.385.3.84 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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 999 of file `basic_string.h`.

Referenced by `std::collate< _CharT >::do_transform()`, `std::operator>>()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`.

```
5.385.3.85 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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 638 of file `basic_string.h`.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

**5.385.3.86** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 629 of file `basic_string.h`.

**5.385.3.87** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 656 of file `basic_string.h`.

**5.385.3.88** `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.

Reimplemented in [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator >](#).

Definition at line 647 of file `basic_string.h`.

**5.385.3.89** `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 1615 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

**5.385.3.90** `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 1547 of file `basic_string.h`.



## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2241

**5.385.3.91** `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 1524 of file `basic_string.h`.

**5.385.3.92** `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 1503 of file `basic_string.h`.

```
5.385.3.93 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 1482 of file `basic_string.h`.

```
5.385.3.94 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.

## 5.385 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference 2243

### 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 1464 of file `basic_string.h`.

Referenced by `std::basic_string<char >::replace()`.

```
5.385.3.95 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 1446 of file `basic_string.h`.

```
5.385.3.96 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 first *n* 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 1423 of file *basic\_string.h*.

```
5.385.3.97 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) [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.

## 5.385 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference 2245

### 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 414 of file `basic_string.tcc`.

```
5.385.3.98 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 1380 of file `basic_string.h`.

Referenced by `std::basic_string<char >::replace()`.

**5.385.3.99** `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.

Definition at line 1358 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

**5.385.3.100** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string<_CharT, _Traits, _Alloc >::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.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference2247

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 502 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>>()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`.

**5.385.3.101** `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 738 of file basic\_string.h.

Referenced by `std::basic_string<char >::resize()`.

**5.385.3.102** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string<_CharT, _Traits, _Alloc >::resize  
(size_type __n, _CharT __c) [inline]`

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 640 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()`.

**5.385.3.103** `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 [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`.

Definition at line 799 of file basic\_string.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.385.3.104** `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 1839 of file basic\_string.h.



## 5.385 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 2249

**5.385.3.105** `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 [inline]`

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 778 of file `basic_string.tcc`.

References `std::min()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.385.3.106** `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 = npow) 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 1811 of file `basic_string.h`.

Referenced by `std::basic_string< char >::rfind()`.

**5.385.3.107** `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 744 of file `basic_string.h`.

**5.385.3.108** `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 700 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::bitset< _S_match_flag_last >::bitset()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `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 >::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< _Ch_type >::transform()`.

**5.385.3.109** `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.

## 5.385 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2251

### 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 2111 of file `basic_string.h`.

**5.385.3.110** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::swap(basic_string<_CharT, _Traits, _Alloc> & __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 519 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::get_allocator()`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, and `std::swap()`.

## 5.385.4 Member Data Documentation

**5.385.4.1** `template<typename _CharT, typename _Traits, typename _Alloc> const basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::npos [inline, static]`

Value returned by various member functions when they fail.

Definition at line 270 of file `basic_string.h`.

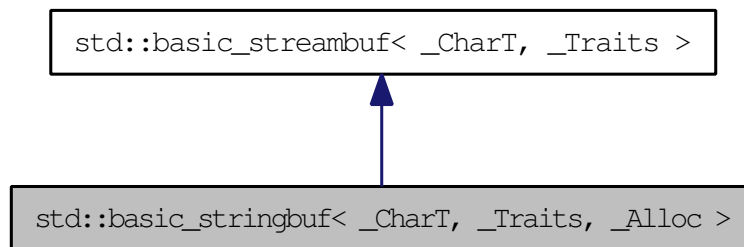
The documentation for this class was generated from the following files:

- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

## 5.386 `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` (`const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `basic_stringbuf` (`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` ()

## 5.386 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

- `int_type` `sputbackc` (`char_type __c`)
- `int_type` `sputc` (`char_type __c`)
- `streamsize` `sputn` (`const char_type *__s, streamsize __n`)
- `void` `stoss` ()
- `void` `str` (`const __string_type &__s`)
- `__string_type` `str` () `const`
- `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` `xspun` (`const char_type *__s, streamsize __n`)

### 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 >`
  
- `locale _M_buf_locale`
- `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 getloc () const`
- `locale pubimbue (const locale &__loc)`
- `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)`
- `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
- `int pubsync ()`
- `char_type * eback () const`

## 5.386 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

- `char_type * egptr () const`
- `char_type * eptr () const`
- `char_type * gptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`

### 5.386.1 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 58 of file `sstream`.

### 5.386.2 Member Typedef Documentation

**5.386.2.1** `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 71 of file `sstream`.

**5.386.2.2** `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 62 of file `sstream`.

**5.386.2.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::int_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::int_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 67 of file sstream.

**5.386.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::off_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::off_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 69 of file sstream.

**5.386.2.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::pos_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::pos_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 68 of file sstream.

**5.386.2.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _Traits std::basic_stringbuf< _CharT, _Traits, _Alloc  
>::traits_type`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 63 of file sstream.



### 5.386.3 Constructor & Destructor Documentation

**5.386.3.1** `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 92 of file sstream.

**5.386.3.2** `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 105 of file sstream.

### 5.386.4 Member Function Documentation

**5.386.4.1** `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 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.386.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr () const [inline,  
protected, 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 466 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.386.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr () const  
[inline, protected, 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 513 of file streambuf.

## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.386.4.4** `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 476 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.386.4.5** `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 222 of file `streambuf`.

**5.386.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::gptr () const [inline,  
protected, 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 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf<_CharT, _Traits >::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf<_CharT, _Traits >::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf<_CharT, _Traits >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::xsgetn()`.

**5.386.4.7** `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 554 of file `streambuf`.

**5.386.4.8** `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.

## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

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 262 of file `streambuf`.

```
5.386.4.9 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()) [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 80 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_stringbuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::eptr()`, `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()`.

```

5.386.4.10 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()) [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 46 of file `sstream.tcc`.

```

5.386.4.11 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 507 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_filebuf< _CharT, _Traits >::sync()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

**5.386.4.12** `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 523 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()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.386.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr () const  
[inline, protected, 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 510 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_filebuf< _CharT, _Traits >::sync()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.386.4.14** `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 205 of file `streambuf`.

**5.386.4.15** `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]`

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 239 of file `streambuf`.

**5.386.4.16** `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]`

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 244 of file `streambuf`.

**5.386.4.17** `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.



## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Definition at line 235 of file streambuf.

**5.386.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_stringbuf< _CharT, _Traits >::pubsync() [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 249 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.386.4.19** `template<typename _CharT, typename _Traits> int_type  
std::basic_stringbuf< _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 294 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

**5.386.4.20** `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) [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 from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 149 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()`.

```
5.386.4.21 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) [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 from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 196 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()`.

```
5.386.4.22 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.

## 5.386 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

### 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 196 of file `sstream`.

References `std::basic_string<_CharT, _Traits, _Alloc>::clear()`.

**5.386.4.23** `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 487 of file `streambuf`.

**5.386.4.24** `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 533 of file `streambuf`.

**5.386.4.25** `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 316 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()`.

**5.386.4.26** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::xsgetn(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 335 of file `streambuf`.

**5.386.4.27** `template<typename _CharT, typename _Traits, typename _Alloc>  
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc  
>::showmanyc() [inline, protected, virtual]`

Investigating the data available.

## 5.386 std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc > Class Template Reference

### 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 164 of file `sstream`.

**5.386.4.28** `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 276 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()`.

**5.386.4.29** `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 350 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.386.4.30** `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 402 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.386.4.31** `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.

### 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.386.4.32** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stoss() [inline,  
inherited]`

Tosses a character. Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.386.4.33** `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 144 of file `sstream`.

References `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.386.4.34** `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 120 of file `sstream`.

**5.386.4.35** `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 375 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.386.4.36** `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 605 of file streambuf.

**5.386.4.37** `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.



## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### 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 678 of file `streambuf`.

```
5.386.4.38 template<class _CharT , class _Traits , class _Alloc >
 basic_stringbuf< _CharT, _Traits, _Alloc >::int_type
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::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 131 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`.

**5.386.4.39** `template<typename _CharT, typename _Traits > streamsize  
std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *  
__s, streamsize __n) [inline, 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 45 of file `streambuf.tcc`.

References `std::min()`.

**5.386.4.40** `template<typename _CharT, typename _Traits > streamsize  
std::basic_streambuf< _CharT, _Traits >::xspunct (const char_type  
* __s, streamsize __n) [inline, 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()`.

## 5.386 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

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 79 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()`.

### 5.386.5 Member Data Documentation

**5.386.5.1** `template<typename _CharT, typename _Traits> locale`  
`std::basic_streambuf<_CharT, _Traits>::_M_buf_locale`  
`[protected, inherited]`

Current `locale` setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

**5.386.5.2** `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 180 of file `streambuf`.

**5.386.5.3** `template<typename _CharT, typename _Traits> char_type*`  
`std::basic_streambuf<_CharT, _Traits>::_M_in_cur`  
`[protected, 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 181 of file `streambuf`.

**5.386.5.4** `template<typename _CharT, typename _Traits> char_type*`  
`std::basic_streambuf< _CharT, _Traits >::_M_in_end`  
`[protected, 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 182 of file `streambuf`.

**5.386.5.5** `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 77 of file `sstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

**5.386.5.6** `template<typename _CharT, typename _Traits> char_type*`  
`std::basic_streambuf< _CharT, _Traits >::_M_out_beg`  
`[protected, 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 183 of file `streambuf`.

## 5.386 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

**5.386.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, 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 184 of file `streambuf`.

**5.386.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, 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 185 of file `streambuf`.

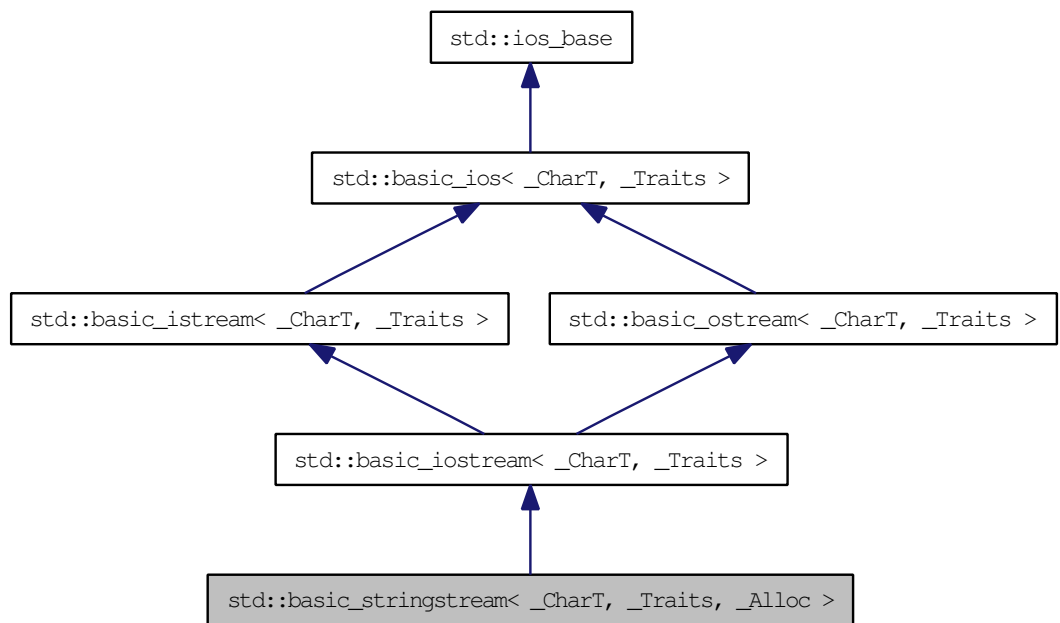
The documentation for this class was generated from the following files:

- [sstream](#)
- [sstream.tcc](#)

## 5.387 `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_istream< char_type, traits_type > __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`
- 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`

## Public Member Functions

- `basic_stringstream` (const `__string_type` &\_\_str, `ios_base::openmode` \_\_m=`ios_base::out|ios_base::in`)
- `basic_stringstream` (`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
- void `exceptions` (`iostate` \_\_except)
- `iostate` `exceptions` () const
- bool `fail` () const
- `char_type` `fill` (`char_type` \_\_ch)
- `char_type` `fill` () const
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- `fmtflags` `flags` () const

- [\\_\\_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 \(streamsize \\_\\_n, int\\_type \\_\\_delim\)](#)
- [template<> basic\\_istream< wchar\\_t > & ignore \(streamsize \\_\\_n\)](#)
- [template<> basic\\_istream< char > & ignore \(streamsize \\_\\_n, int\\_type \\_\\_delim\)](#)
- [template<> basic\\_istream< char > & ignore \(streamsize \\_\\_n\)](#)
- [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 \(basic\\_streambuf< \\_CharT, \\_Traits > \\*\\_\\_sb\)](#)
- [\\_\\_stringbuf\\_type \\* rdbuf \(\) const](#)
- [iostate 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 \(fmtflags \\_\\_fmtfl, fmtflags \\_\\_mask\)](#)
- [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 \(streamsize \\_\\_wide\)](#)



- [streamsize width \(\)](#) const

### 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.

- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### 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.

- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type &__c)`
- `int_type get ()`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & putback (char_type __c)`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`

- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__istream_type & seekg (pos_type)`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & unget ()`

### 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<< (__streambuf_type *__sb)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`

### 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>> (__streambuf_type * __sb)`
- `__istream_type & operator>> (void *& __p)`
- `__istream_type & operator>> (long double & __f)`
- `__istream_type & operator>> (double & __f)`
- `__istream_type & operator>> (float & __f)`
- `__istream_type & operator>> (unsigned long long & __n)`
- `__istream_type & operator>> (long long & __n)`
- `__istream_type & operator>> (unsigned long & __n)`
- `__istream_type & operator>> (long & __n)`
- `__istream_type & operator>> (unsigned int & __n)`
- `__istream_type & operator>> (int & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (bool & __n)`
  
- `__istream_type & operator>> (ios_base & (*__pf)(ios_base &))`
- `__istream_type & operator>> (__ios_type & (*__pf)(__ios_type &))`
- `__istream_type & operator>> (__istream_type & (*__pf)(__istream_type &))`

### 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 [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](#)
- [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`
- `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`
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> __num_put_type`
- `operator void * () const`
- `bool operator! () const`

### 5.387.1 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 476 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 71 of file ostream.

**5.387.2.2** `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 71 of file istream.

**5.387.2.3** `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 70 of file istream.

**5.387.2.4** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream< _CharT, _Traits >::_num_put_type  
[inherited]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.387.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::_num_put_type [inherited]`

The quick-and-easy status check. This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

**5.387.2.6** `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 480 of file `sstream`.

**5.387.2.7** `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 444 of file `ios_base.h`.

**5.387.2.8** `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 263 of file ios\_base.h.

**5.387.2.9** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef traits_type::int_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are non-standard types.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 485 of file sstream.



**5.387.2.10 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 338 of file *ios\_base.h*.

**5.387.2.11 template<typename \_CharT, typename \_Traits, typename \_Alloc > typedef traits\_type::off\_type std::basic\_stringstream<\_CharT, \_Traits, \_Alloc >::off\_type**

These are non-standard types.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits >](#).

Definition at line 487 of file *sstream*.

**5.387.2.12 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 369 of file *ios\_base.h*.

**5.387.2.13** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef traits_type::pos_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::pos_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 486 of file `sstream`.

**5.387.2.14** `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 401 of file `ios_base.h`.

**5.387.2.15** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef _Traits std::basic_stringstream< _CharT, _Traits, _Alloc  
>::traits_type`

These are non-standard types.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 481 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 427 of file `ios_base.h`.

## 5.387.4 Constructor & Destructor Documentation

**5.387.4.1** `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.

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 510 of file `sstream`.

**5.387.4.2** `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 526 of file `sstream`.

**5.387.4.3** `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 537 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 705 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 287 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.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2293

### Returns:

True if the badbit is [set](#).

Note that other iostate flags may also be [set](#).

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

```
5.387.5.4 template<typename _CharT , typename _Traits > void
std::basic_ios<_CharT, _Traits >::clear (iostate __state = goodbit)
[inline, 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 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits >::rdbuf()`.

```
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) [inline,
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 62 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_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 180 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()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

### 5.387.5.7 `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 247 of file `basic_ios.h`.

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2295

---

**5.387.5.8** `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 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**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 191 of file `basic_ios.h`.

Referenced by `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(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 380 of file `basic_ios.h`.

**5.387.5.11** `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 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.387.5.12** `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 561 of file `ios_base.h`.

**5.387.5.13** `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

**Returns:**

The format control flags for both input and output.



## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2297

Definition at line 550 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<<()`, and `std::operator>>()`.

### 5.387.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush() [inline, 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 211 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 `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 249 of file `istream`.

### 5.387.5.16 `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 366 of file istream.

**5.387.5.17** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::get(_streambuf_type & __sb, char_type __delim) [inline, 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 356 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()`.

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2299

---

**5.387.5.18** `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 333 of file istream.

**5.387.5.19** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get(char_type * __s, streamsize __n, char_type __delim) [inline, 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 309 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()`.

**5.387.5.20** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get(char_type & __c) [inline, 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 272 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()`.

**5.387.5.21** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get(void) [inline, inherited]`

Simple extraction.

**Returns:**

A character, or `eof()`.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2301

---

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 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()`.

**5.387.5.22** `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 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.387.5.23** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline(char_type * __s, streamsize __n, char_type __delim) [inline, 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 400 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()`.

#### 5.387.5.24 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 694 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()`.

#### 5.387.5.25 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good() const` [[inline](#), [inherited](#)]

Fast error checking.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2303

### Returns:

True if no error flags are [set](#).

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.387.5.26** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim) [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 555 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()`.

**5.387.5.27** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n) [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 493 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()`.

**5.387.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (void) [inline, 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  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  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 460 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()`.

**5.387.5.29** `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc) [inline, inherited]`

Moves to a new [locale](#).



## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2305

### 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 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.387.5.30** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init(basic_streambuf<_CharT,  
_Traits> * __sb) [inline, 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 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.387.5.31** `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 740 of file ios\_base.h.

**5.387.5.32** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file basic\_ios.h.

**5.387.5.33** `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 111 of file basic\_ios.h.

**5.387.5.34** `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 115 of file basic\_ios.h.

**5.387.5.35** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (_streambuf_type * __sb) [inline, 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 120 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()`.

**5.387.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (const void * __p) [inline, 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 225 of file ostream.

**5.387.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (long double __f) [inline, 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 221 of file ostream.

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference 2309**

---

**5.387.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (float __f) [inline, 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 213 of file ostream.

**5.387.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (double __f) [inline, 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 209 of file ostream.

**5.387.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline, 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 204 of file ostream.

**5.387.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline, inherited]`

Extracting from another streambuf.

**Parameters:**

*sb* A pointer to a streambuf

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference 2311**

---

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 200 of file ostream.

**5.387.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned int __n) [inline, 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 191 of file ostream.

**5.387.5.43** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<< (int __n) [inline, 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 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.387.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned short __n) [inline, 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 180 of file ostream.

**5.387.5.45** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (short __n) [inline, 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 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.387.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (bool __n) [inline, 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 173 of file ostream.

```
5.387.5.47 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits >::operator<< (unsigned
long __n) [inline, 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

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2315

---

If the function inserts no characters, failbit is [set](#).

Definition at line 169 of file ostream.

**5.387.5.48** `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 165 of file ostream.

**5.387.5.49** `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 127 of file ostream.

**5.387.5.50** `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 117 of file ostream.

**5.387.5.51** `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 `omanip` header.

Definition at line 108 of file `ostream`.

**5.387.5.52** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * sb) [inline, 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 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_istream< _CharT, _Traits >::rdbuf()`, and `std::basic_istream< _CharT, _Traits >::setstate()`.

**5.387.553** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (void *& __p) [inline, 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 215 of file istream.

**5.387.554** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long double & __f) [inline, 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 211 of file istream.

```
5.387.5.55 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (double & __f)
[inline, 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 207 of file istream.

```
5.387.5.56 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (float & __f)
[inline, inherited]
```

Extracting into another streambuf.

**Parameters:**

*sb* A pointer to a streambuf

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference 2319**

---

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 203 of file istream.

**5.387.5.57 template<typename \_CharT, typename \_Traits> \_\_istream\_type& std::basic\_istream<\_CharT, \_Traits >::operator>> (unsigned long long & \_\_n) [inline, 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 198 of file istream.

**5.387.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline, 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 194 of file istream.

**5.387.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline, 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,



## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2321

- 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 189 of file `istream`.

**5.387.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long & __n) [inline, 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 185 of file `istream`.

**5.387.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned int & __n) [inline, 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 181 of file istream.

```
5.387.5.62 template<typename _CharT , typename _Traits > basic_istream<
 _CharT, _Traits > & std::basic_istream< _CharT, _Traits
 >::operator>> (int & __n) [inline, 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 159 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\(\)](#).

**5.387.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned short & __n) [inline, 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 174 of file `istream`.

**5.387.5.64** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (short & __n) [inline, 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 114 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()`.

**5.387.5.65** `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 167 of file istream.

**5.387.5.66** `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 131 of file istream.

**5.387.5.67** `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 124 of file `istream`.

**5.387.5.68** `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 `iomanip` header.

Definition at line 120 of file `istream`.

**5.387.5.69** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek(void)` [`inline`, `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 620 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()`.

**5.387.5.70** `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 629 of file `ios_base.h`.

**5.387.5.71** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.387.5.72** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put(char_type __c) [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 149 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()`.

**5.387.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback(char_type __c) [inline, inherited]`

Unextracting a single character.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2327

### 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 711 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

### 5.387.5.74 `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 761 of file `ios_base.h`.

**5.387.5.75** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf(basic_streambuf<_CharT, _Traits> * __sb) [inline, 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 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.387.5.76** `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 548 of file `sstream`.



**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2329

---

**5.387.5.77** `template<typename _CharT, typename _Traits> istate  
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 127 of file `basic_ios.h`.

**5.387.5.78** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read  
(char_type * __s, streamsize __n) [inline, 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 650 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()`.

**5.387.5.79** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome(char_type * __s,  
streamsize __n) [inline, 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 679 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()`.

**5.387.5.80** `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.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template

Reference

2331

**5.387.5.81** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg(off_type __off, ios_base::seekdir __dir) [inline, 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 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.387.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg(pos_type __pos) [inline, 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 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.387.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inline, 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 290 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()`.

**5.387.5.84** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inline, 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 258 of file ostream.tcc.

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2333

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()`.

### 5.387.5.85 `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 594 of file `ios_base.h`.

### 5.387.5.86 `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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.387.5.87** `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 147 of file `basic_ios.h`.

Referenced by `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_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 >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.387.5.88** `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 `rdbuf ()->str (s)`.

Definition at line 566 of file `sstream`.

**5.387.5.89** `template<typename _CharT , typename _Traits , typename _Alloc  
> __string_type std::basic_stringstream< _CharT, _Traits, _Alloc  
>::str () const [inline]`

Copying out the string buffer.

**Returns:**

`rdbuf ()->str ()`

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2335

Definition at line 556 of file `sstream`.

```
5.387.5.90 template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void) [inline,
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 777 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()`.

```
5.387.5.91 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>

**5.387.5.92** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inline, 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 813 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()`.

**5.387.5.93** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inline, 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 237 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()`.

**5.387.5.94** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tistr) [inline, inherited]`

Ties this stream to an output stream.



## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2337

### 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 297 of file `basic_ios.h`.

```
5.387.5.95 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 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

```
5.387.5.96 template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inline, 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 744 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::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.387.5.97** `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 609 of file ios\_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.387.5.98** `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 I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic\_ios.h.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference** 2339

---

**5.387.5.99** `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 652 of file `ios_base.h`.

**5.387.5.100** `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 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::operator<>>()`.

**5.387.5.101** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::write (const char_type * __s, streamsize __n) [inline, 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 183 of file `ostream.tcc`.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.387.5.102 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.

**5.387.6 Member Data Documentation****5.387.6.1 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 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`,

## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2341

---

`std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.387.6.2 `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 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

### 5.387.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

### 5.387.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

### 5.387.6.5 `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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::basic_ostream<_CharT, _Traits>::write()`.

**5.387.6.6 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 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.387.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::seekpos()`.

**5.387.6.8 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 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

**5.387.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

**5.387.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `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 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2343

### 5.387.6.11 `const fmtflags std::ios_base::dec` [static, inherited]

Converts integer input or generates integer output in [decimal](#) base.

Definition at line 269 of file `ios_base.h`.

### 5.387.6.12 `const seekdir std::ios_base::end` [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

### 5.387.6.13 `const iostate std::ios_base::eofbit` [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::ws()`.

### 5.387.6.14 `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 350 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_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `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()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

**5.387.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.387.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios\_base.h.

**5.387.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 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 >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.387.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.387.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.



## 5.387 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2345

Referenced by `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_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()`.

### 5.387.6.20 `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 280 of file `ios_base.h`.

### 5.387.6.21 `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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter>::do_put()`.

### 5.387.6.22 `const fmtflags std::ios_base::oct` [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

### 5.387.6.23 `const openmode std::ios_base::out` [static, inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `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()`.

**5.387.6.24 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 291 of file ios\_base.h.

**5.387.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.387.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file ios\_base.h.

**5.387.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file ios\_base.h.

**5.387.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.387.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.387.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.387 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference** 2347

---

**5.387.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.387.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

The documentation for this class was generated from the following file:

- [sstream](#)

## 5.388 `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](#) (const [param\\_type](#) &\_\_p)
- [bernoulli\\_distribution](#) (double \_\_p=0.5)
- [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](#) ()

#### 5.388.1 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 2868 of file random.h.

#### 5.388.2 Member Typedef Documentation

##### 5.388.2.1 typedef bool `std::bernoulli_distribution::result_type`

The type of the range of the distribution.

Definition at line 2872 of file random.h.

### 5.388.3 Constructor & Destructor Documentation

**5.388.3.1** `std::bernoulli_distribution::bernoulli_distribution (double __p = 0.5) [inline, explicit]`

Constructs a Bernoulli distribution with likelihood  $p$ .

**Parameters:**

$\text{__}p$  [IN] The likelihood of a true result being returned. Must be in the interval  $[0, 1]$ .

Definition at line 2901 of file random.h.

### 5.388.4 Member Function Documentation

**5.388.4.1** `result_type std::bernoulli_distribution::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2951 of file random.h.

**5.388.4.2** `result_type std::bernoulli_distribution::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2944 of file random.h.

**5.388.4.3** `template<typename _UniformRandomNumberGenerator > result_type std::bernoulli_distribution::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Returns the next value in the Bernoullian sequence.

Definition at line 2959 of file random.h.

References operator(), and param().

Referenced by operator().

**5.388.4.4** `double std::bernoulli_distribution::p () const [inline]`

Returns the  $p$  parameter of the distribution.

Definition at line 2922 of file random.h.

**5.388.4.5** `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 2937 of file random.h.

**5.388.4.6** `param_type std::bernoulli_distribution::param () const [inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 2929 of file random.h.

Referenced by `operator()()`, and `std::operator>>()`.

**5.388.4.7** `void std::bernoulli_distribution::reset () [inline]`

Resets the distribution state. Does nothing for a Bernoulli distribution.

Definition at line 2916 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.389 `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

#### 5.389.1 Detailed Description

Parameter type.

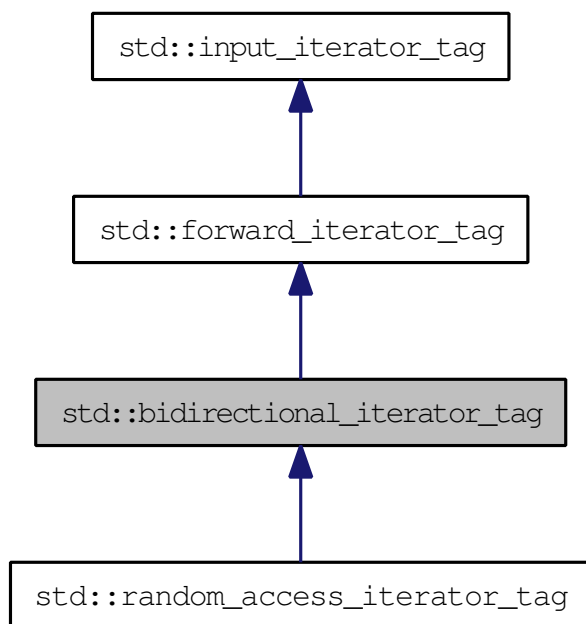
Definition at line 2874 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.390 `std::bidirectional_iterator_tag` Struct Reference

Bidirectional iterators support a superset of forward [iterator](#) operations. Inheritance diagram for `std::bidirectional_iterator_tag`:



### 5.390.1 Detailed Description

Bidirectional iterators support a superset of forward [iterator](#) operations.

Definition at line 89 of file `stl_iterator_base_types.h`.

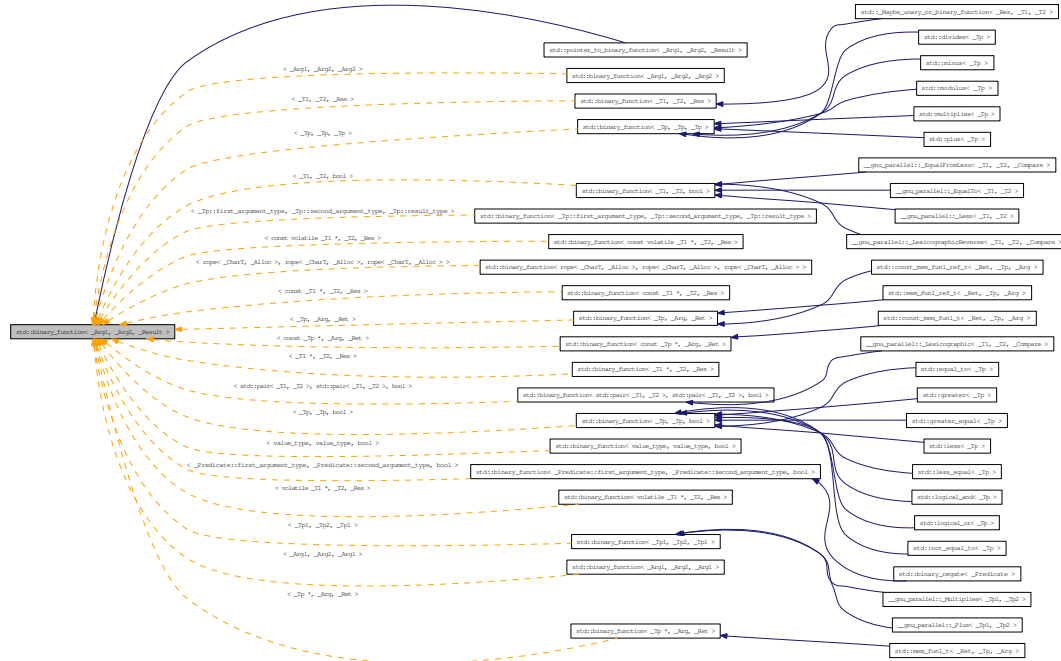
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)



## 5.391 std::binary\_function< \_Arg1, \_Arg2, \_Result > Struct Template Reference

Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



### Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

### 5.391.1 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 112 of file stl\_function.h.

## 5.391.2 Member Typedef Documentation

**5.391.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result  
>::first_argument_type`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.391.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Result std::binary_function< _Arg1, _Arg2, _Result  
>::result_type`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.391.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result  
>::second_argument_type`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.392 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`

#### 5.392.1 Detailed Description

```
template<typename _Predicate> class std::binary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 369 of file `stl_function.h`.

#### 5.392.2 Member Typedef Documentation

**5.392.2.1** `typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type` `[inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.392.2.2** `typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.392.2.3** `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]`

the type of the second argument

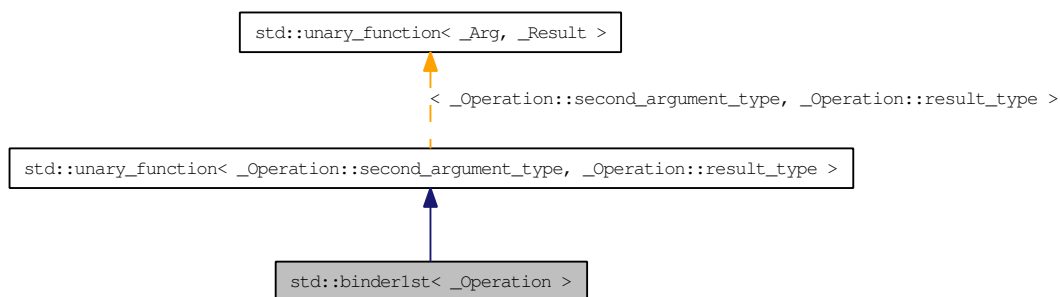
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.393 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**

#### 5.393.1 Detailed Description

```
template<typename _Operation> class std::binder1st< _Operation >
```

One of the [binder functors](#).

Definition at line 98 of file `binders.h`.

## 5.393.2 Member Typedef Documentation

**5.393.2.1** `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 (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.393.2.2** `typedef _Operation::result_type std::unary_function<_Operation::second_argument_type, _Operation::result_type>::result_type [inherited]`

`result_type` is the return type

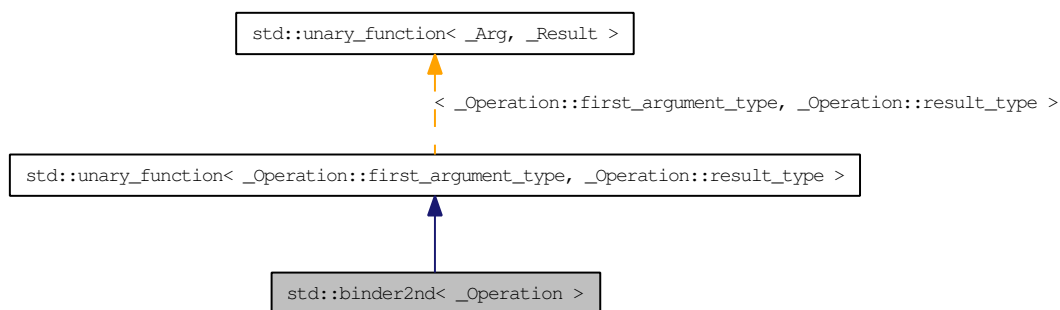
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.394 `std::binder2nd< _Operation >` Class Template Reference

One of the [binder functions](#). 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**

#### 5.394.1 Detailed Description

```
template<typename _Operation> class std::binder2nd< _Operation >
```

One of the [binder functions](#).

Definition at line 133 of file `binders.h`.

## 5.394.2 Member Typedef Documentation

**5.394.2.1** `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 (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.394.2.2** `typedef _Operation::result_type std::unary_function<_Operation::first_argument_type , _Operation::result_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)



## 5.395 `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** (const [param\\_type](#) &\_\_p)
- **binomial\_distribution** (`_IntType` \_\_t=`_IntType`(1), double \_\_p=0.5)
- [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`, typename `_CharT`, typename `_Traits` >  
`std::basic_istream<_CharT, _Traits>` & **operator>>** (`std::basic_istream<_CharT, _Traits>` &, `std::binomial_distribution<_IntType1>` &)

### 5.395.1 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  $p(i|t, p) = \binom{t}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3023 of file random.h.

### 5.395.2 Member Typedef Documentation

**5.395.2.1** `template<typename _IntType = int> typedef _IntType  
std::binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3030 of file random.h.

### 5.395.3 Member Function Documentation

**5.395.3.1** `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 3129 of file random.h.

Referenced by `std::binomial_distribution< _IntType >::operator()()`.

**5.395.3.2** `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 3122 of file random.h.

**5.395.3.3** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > binomial_distribution<  
_IntType >::result_type std::binomial_distribution<_IntType  
>::operator() (_UniformRandomNumberGenerator & __urng, const  
param_type & __param) [inline]`

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 1380 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::binomial_distribution<_IntType>::max()`.

**5.395.3.4** `template<typename _IntType = int> double  
std::binomial_distribution<_IntType >::p () const [inline]`

Returns the distribution `p` parameter.

Definition at line 3100 of file `random.h`.

**5.395.3.5** `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 3115 of file `random.h`.

**5.395.3.6** `template<typename _IntType = int> param_type  
std::binomial_distribution<_IntType >::param () const [inline]`

Returns the parameter `set` of the distribution.

Definition at line 3107 of file `random.h`.

**5.395.3.7** `template<typename _IntType = int> void  
std::binomial_distribution<_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 3086 of file random.h.

References `std::normal_distribution<_RealType>::reset()`.

**5.395.3.8** `template<typename _IntType = int> _IntType  
std::binomial_distribution<_IntType>::t() const [inline]`

Returns the distribution `t` parameter.

Definition at line 3093 of file random.h.

## 5.395.4 Friends And Related Function Documentation

**5.395.4.1** `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.

**5.395.4.2** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> (std::basic_istream<_CharT, _Traits > &,  
std::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.

### Returns:

The input stream with `__x` extracted or in an error state.

## **5.395 std::binomial\_distribution<\_IntType> Class Template Reference 2365**

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.396 `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>](#)

#### 5.396.1 Detailed Description

`template<typename _IntType = int> struct std::binomial_distribution<_IntType>::param_type`

Parameter type.

Definition at line 3032 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.397 `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

- [bitset](#) (const char \* \_\_str)
- `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, size\_t \_\_n)
- `template<class _CharT, class _Traits, class _Alloc >`  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position=0)
- [bitset](#) (unsigned long long \_\_val)
- [bitset](#) ()
- 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)
- `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\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s) const
- `template<class _CharT, class _Traits, class _Alloc >`  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &, \_CharT, \_CharT) const

- bool [all](#) () const
  - bool [any](#) () const
  - [size\\_t](#) [count](#) () const
  - [bitset](#)< [\\_Nb](#) > & [flip](#) ([size\\_t](#) \_\_position)
  - [bitset](#)< [\\_Nb](#) > & [flip](#) ()
  - bool [none](#) () const
  - [bitset](#)< [\\_Nb](#) > [operator~](#) () const
  - [bitset](#)< [\\_Nb](#) > & [reset](#) ([size\\_t](#) \_\_position)
  - [bitset](#)< [\\_Nb](#) > & [reset](#) ()
  - [bitset](#)< [\\_Nb](#) > & [set](#) ([size\\_t](#) \_\_position, bool \_\_val=true)
  - [bitset](#)< [\\_Nb](#) > & [set](#) ()
  - [size\\_t](#) [size](#) () const
  - bool [test](#) ([size\\_t](#) \_\_position) 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
  - [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](#) >  
[std::basic\\_string](#)< [\\_CharT](#), [std::char\\_traits](#)< [\\_CharT](#) >, [std::allocator](#)< [\\_CharT](#) > > [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](#) >  
[std::basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [std::allocator](#)< [\\_CharT](#) > > [to\\_string](#) () const
  - [template](#)<class [\\_CharT](#), class [\\_Traits](#), class [\\_Alloc](#) >  
[std::basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Alloc](#) > [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
  - unsigned long long [to\\_ullong](#) () const
  - unsigned long [to\\_ulong](#) () const
- 
- [bitset](#)< [\\_Nb](#) > & [\\_Unchecked\\_flip](#) ([size\\_t](#) \_\_pos)
  - [bitset](#)< [\\_Nb](#) > & [\\_Unchecked\\_reset](#) ([size\\_t](#) \_\_pos)
  - [bitset](#)< [\\_Nb](#) > & [\\_Unchecked\\_set](#) ([size\\_t](#) \_\_pos, int \_\_val)
  - [bitset](#)< [\\_Nb](#) > & [\\_Unchecked\\_set](#) ([size\\_t](#) \_\_pos)
  - bool [\\_Unchecked\\_test](#) ([size\\_t](#) \_\_pos) const
  - bool [operator!=](#) (const [bitset](#)< [\\_Nb](#) > & \_\_rhs) const
  - [bitset](#)< [\\_Nb](#) > & [operator&=](#) (const [bitset](#)< [\\_Nb](#) > & \_\_rhs)



- `bitset< _Nb > operator<< (size_t __position) const`
- `bitset< _Nb > & operator<<= (size_t __position)`
- `bool operator== (const bitset< _Nb > &__rhs) const`
- `bitset< _Nb > operator>> (size_t __position) const`
- `bitset< _Nb > & operator>>= (size_t __position)`
- `bool operator[] (size_t __position) const`
- `reference operator[] (size_t __position)`
- `bitset< _Nb > & operator^= (const bitset< _Nb > &__rhs)`
- `bitset< _Nb > & operator|= (const bitset< _Nb > &__rhs)`

## Private Types

- typedef unsigned long `_WordT`

## 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) const`
- `_WordT & _M_getword (size_t __pos)`
- `_WordT _M_hiword () const`
- `_WordT & _M_hiword ()`
- `bool _M_is_any () const`
- `bool _M_is_equal (const _Base_bitset< _Nw > &__x) const`

## Static Private Member Functions

- `static _WordT _S_maskbit (size_t __pos)`
- `static size_t _S_whichbit (size_t __pos)`
- `static size_t _S_whichbyte (size_t __pos)`
- `static size_t _S_whichword (size_t __pos)`

## Private Attributes

- `_WordT` `_M_w` [`_Nw`]

## Friends

- class `hash`
- class `reference`

### 5.397.1 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 709 of file `bitset`.

## 5.397.2 Constructor & Destructor Documentation

### 5.397.2.1 `template<size_t _Nb> std::bitset<_Nb>::bitset () [inline]`

All bits `set` to zero.

Definition at line 803 of file `bitset`.

### 5.397.2.2 `template<size_t _Nb> std::bitset<_Nb>::bitset (unsigned long long __val) [inline]`

Initial bits bitwise-copied from a single word (others `set` to zero).

Definition at line 808 of file `bitset`.

### 5.397.2.3 `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 826 of file `bitset`.

**5.397.2.4** `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 848 of file `bitset`.

**5.397.2.5** `template<size_t _Nb> std::bitset<_Nb>::bitset (const char * __str) [inline, explicit]`

Construct from a string.

**Parameters:**

*str* A string of *0* and *1* characters.

**Exceptions:**

*std::invalid\_argument* If a character appears in the string which is neither *0* nor *1*.

Definition at line 880 of file `bitset`.

### 5.397.3 Member Function Documentation

**5.397.3.1** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::_Unchecked_flip (size_t __pos) [inline]`

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

Definition at line 987 of file `bitset`.

**5.397.3.2** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::_Unchecked_reset (size_t __pos) [inline]`

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

Definition at line 980 of file `bitset`.

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

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

Definition at line 970 of file `bitset`.

**5.397.3.4** `template<size_t _Nb> bool std::bitset< _Nb >::_Unchecked_test (size_t __pos) const [inline]`

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

Definition at line 994 of file `bitset`.

**5.397.3.5** `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 1267 of file `bitset`.

**5.397.3.6** `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 1275 of file `bitset`.

**5.397.3.7** `template<size_t _Nb> size_t std::bitset< _Nb >::count () const`  
`[inline]`

Returns the number of bits which are [set](#).

Definition at line 1227 of file `bitset`.

**5.397.3.8** `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 1067 of file `bitset`.

**5.397.3.9** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::flip ()`  
`[inline]`

Toggles every bit to its opposite value.

Definition at line 1054 of file `bitset`.

**5.397.3.10** `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 1283 of file `bitset`.

**5.397.3.11** `template<size_t _Nb> bool std::bitset< _Nb >::operator!=(const bitset< _Nb > & __rhs) const [inline]`

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

Definition at line 1242 of file `bitset`.

**5.397.3.12** `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 901 of file `bitset`.

**5.397.3.13** `template<size_t _Nb> bitset<_Nb> std::bitset< _Nb >::operator<< (size_t __position) const [inline]`

Self-explanatory.

Definition at line 1289 of file `bitset`.

**5.397.3.14** `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 930 of file `bitset`.

**5.397.3.15** `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 1238 of file `bitset`.

**5.397.3.16** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb >::operator>>(size_t __position) const [inline]`

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

Definition at line 1293 of file `bitset`.

**5.397.3.17** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::operator>>=(size_t __position) [inline]`

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

Definition at line 943 of file `bitset`.

**5.397.3.18** `template<size_t _Nb> bool std::bitset<_Nb >::operator[] (size_t __position) const [inline]`

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

Definition at line 1099 of file `bitset`.

**5.397.3.19** `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 1095 of file `bitset`.

**5.397.3.20** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator^=(const bitset<_Nb> & __rhs) [inline]`

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

Definition at line 915 of file `bitset`.

**5.397.3.21** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator|=(const bitset<_Nb> & __rhs) [inline]`

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

Definition at line 908 of file `bitset`.

**5.397.3.22** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb>::operator~() const [inline]`

See the no-argument `flip()`.

Definition at line 1076 of file `bitset`.

**5.397.3.23** `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 1043 of file `bitset`.

**5.397.3.24** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::reset ()`  
`[inline]`

Sets every bit to false.

Definition at line 1029 of file `bitset`.

**5.397.3.25** `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 1018 of file `bitset`.

**5.397.3.26** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::set ()`  
`[inline]`

Sets every bit to true.

Definition at line 1004 of file `bitset`.

**5.397.3.27** `template<size_t _Nb> size_t std::bitset<_Nb >::size () const`  
`[inline]`

Returns the total number of bits.

Definition at line 1232 of file `bitset`.

**5.397.3.28** `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 1253 of file `bitset`.

**5.397.3.29** `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 1129 of file `bitset`.

**5.397.3.30** `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 1110 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

## 5.398 `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`

### 5.398.1 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 740 of file `bitset`.

The documentation for this class was generated from the following file:

- `bitset`

## 5.399 `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](#) (const [param\\_type](#) &\_\_p)
- [cauchy\\_distribution](#) (`_RealType` \_\_a=`_RealType`(0), `_RealType` \_\_b=`_RealType`(1))
- `_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](#) ()

#### 5.399.1 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 2416 of file random.h.

## 5.399.2 Member Typedef Documentation

### 5.399.2.1 `template<typename _RealType = double> typedef _RealType std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2423 of file random.h.

## 5.399.3 Member Function Documentation

### 5.399.3.1 `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 2503 of file random.h.

### 5.399.3.2 `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 2496 of file random.h.

### 5.399.3.3 `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 2489 of file random.h.

### 5.399.3.4 `template<typename _RealType = double> param_type std::cauchy_distribution< _RealType >::param () const [inline]`

Returns the parameter `set` of the distribution.

Definition at line 2481 of file random.h.

Referenced by `std::operator>>()`.

## 5.399 std::cauchy\_distribution<\_RealType > Class Template Reference 2383

**5.399.3.5** `template<typename _RealType = double> void  
std::cauchy_distribution<_RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2463 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.400 `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

#### 5.400.1 Detailed Description

`template<typename _RealType = double> struct std::cauchy_distribution<_RealType >::param_type`

Parameter type.

Definition at line 2425 of file `random.h`.

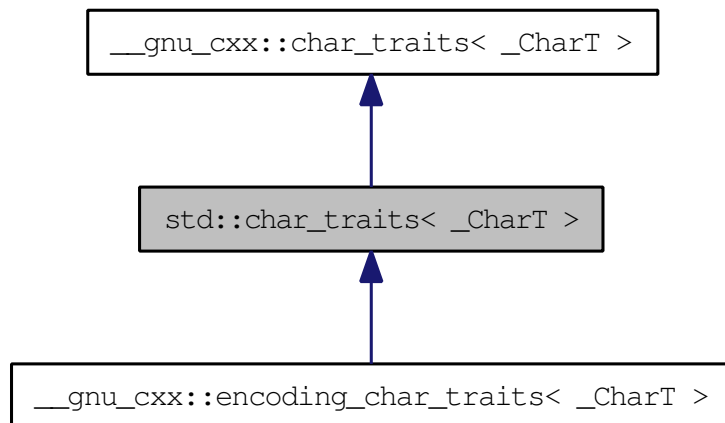
The documentation for this struct was generated from the following file:

- [random.h](#)



## 5.401 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 char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static void **assign** (char\_type & \_\_c1, const char\_type & \_\_c2)
- 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 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, std::size\_t \_\_n, const char\_type & \_\_a)

- static `std::size_t` **length** (`const char_type *__s`)
- static `bool` **It** (`const char_type &__c1, const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1, const char_type *__s2, std::size_t __n`)
- static `int_type` **not\_eof** (`const int_type &__c`)
- static `char_type` **to\_char\_type** (`const int_type &__c`)
- static `int_type` **to\_int\_type** (`const char_type &__c`)

### 5.401.1 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 231 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.402 `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 `char_type * assign` (`char_type * __s`, `size_t __n`, `char_type __a`)
- static void `assign` (`char_type & __c1`, const `char_type & __c2`)
- 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`)

### 5.402.1 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 88 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.403 `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 char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- 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 &\_\_c)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.403.1 Detailed Description

`template<> struct std::char_traits< char >`

21.1.3.1 [char\\_traits](#) specializations

Definition at line 237 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.404 `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 `char_type` \* **assign** (`char_type` \*\_\_s, `size_t` \_\_n, `char_type` \_\_a)
- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- 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` &\_\_c)
- static `int_type` **to\_int\_type** (const `char_type` &\_\_c)

### 5.404.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

21.1.3.2 `char_traits` specializations

Definition at line 308 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.405 `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` (const [param\\_type](#) &\_\_p)
- `chi_squared_distribution` (`_RealType` \_\_n=`_RealType`(1))
- `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::chi_squared_distribution`< `_RealType1` > &)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & [operator>>](#) (`std::basic_istream`< `_CharT`, `_Traits` > &, `std::chi_squared_distribution`< `_RealType1` > &)

## 5.405 `std::chi_squared_distribution<_RealType>` Class Template Reference 2391

### 5.405.1 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 2278 of file `random.h`.

### 5.405.2 Member Typedef Documentation

**5.405.2.1** `template<typename _RealType = double> typedef _RealType std::chi_squared_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2285 of file `random.h`.

### 5.405.3 Member Function Documentation

**5.405.3.1** `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 2354 of file `random.h`.

**5.405.3.2** `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 2347 of file `random.h`.

**5.405.3.3** `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 2340 of file random.h.

**5.405.3.4** `template<typename _RealType = double> param_type  
std::chi_squared_distribution<_RealType >::param () const  
[inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 2332 of file random.h.

**5.405.3.5** `template<typename _RealType = double> void  
std::chi_squared_distribution<_RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2318 of file random.h.

References `std::gamma_distribution<_RealType >::reset()`.

**5.405.4 Friends And Related Function Documentation**

**5.405.4.1** `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.



## 5.405 std::chi\_squared\_distribution<\_RealType> Class Template Reference 2393

**5.405.4.2** `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> &, std::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](#)

## 5.406 `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

#### 5.406.1 Detailed Description

`template<typename _RealType = double> struct std::chi_squared_distribution<_RealType>::param_type`

Parameter type.

Definition at line 2287 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.407 `std::chrono::duration< _Rep, _Period >` Struct Template Reference

[duration](#)

### Public Types

- typedef `_Period` **period**
- typedef `_Rep` **rep**

### Public Member Functions

- **duration** (const [duration](#) &)
- 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>  
**duration** (const [duration](#)< `_Rep2`, `_Period2` > &\_\_d)
- 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>  
**duration** (const `_Rep2` &\_\_rep)
- `rep` **count** () const
- template<typename `_Rep2` = rep>  
[enable\\_if](#)<!treat\_as\_floating\_point< `_Rep2` >::value, [duration](#) & >::type **operator** %= (const [duration](#) &\_\_d)
- template<typename `_Rep2` = rep>  
[enable\\_if](#)<!treat\_as\_floating\_point< `_Rep2` >::value, [duration](#) & >::type **operator** %= (const rep &\_\_rhs)
- [duration](#) & **operator** \*= (const rep &\_\_rhs)
- [duration](#) **operator** + () const
- [duration](#) **operator** ++ (int)
- [duration](#) & **operator** ++ ()
- [duration](#) & **operator** += (const [duration](#) &\_\_d)
- [duration](#) **operator** - () const
- [duration](#) **operator** -- (int)
- [duration](#) & **operator** -- ()
- [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_ratio< _Period >::value`, "period must be a specialization of [ratio](#)")
- **static\_assert** (!`__is_duration< _Rep >::value`, "rep cannot be a [duration](#)")

## Static Public Member Functions

- static const [duration](#) **max** ()
- static const [duration](#) **min** ()
- static const [duration](#) **zero** ()

### 5.407.1 Detailed Description

**template**<typename **\_Rep**, typename **\_Period**> **struct** `std::chrono::duration`< **\_Rep**, **\_Period** >

[duration](#)

Definition at line 201 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## **5.408** `std::chrono::duration_values<_Rep>` Struct Template Reference

[duration\\_values](#)

### **Static Public Member Functions**

- static const `_Rep` **max** ()
- static const `_Rep` **min** ()
- static const `_Rep` **zero** ()

### **5.408.1 Detailed Description**

`template<typename _Rep> struct std::chrono::duration_values<_Rep>`

[duration\\_values](#)

Definition at line 174 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.409 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**

### 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)

### Static Public Attributes

- static const bool **is\_monotonic**

#### 5.409.1 Detailed Description

[system\\_clock](#)

Definition at line 628 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.410 `std::chrono::time_point< _Clock, _Duration >` Struct Template Reference

[time\\_point](#)

### Public Types

- typedef `_Clock` **clock**
- typedef `_Duration` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

### Public Member Functions

- `template<typename _Duration2 >`  
**time\_point** (`const time\_point< clock, _Duration2 > &__t`)
- **time\_point** (`const duration &__dur`)
- **time\_point** & **operator+=** (`const duration &__dur`)
- **time\_point** & **operator-=** (`const duration &__dur`)
- `duration` **time\_since\_epoch** () const

### Static Public Member Functions

- static const **time\_point** **max** ()
- static const **time\_point** **min** ()

#### 5.410.1 Detailed Description

`template<typename _Clock, typename _Duration> struct std::chrono::time_point< _Clock, _Duration >`

[time\\_point](#)

Definition at line 492 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.411 `std::chrono::treat_as_floating_point< _Rep >` Struct Template Reference

[treat\\_as\\_floating\\_point](#) Inheritance diagram for `std::chrono::treat_as_floating_point< _Rep >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Static Public Attributes

- static const bool **value**

#### 5.411.1 Detailed Description

`template<typename _Rep> struct std::chrono::treat_as_floating_point< _Rep >`

[treat\\_as\\_floating\\_point](#)

Definition at line 168 of file `chrono`.

The documentation for this struct was generated from the following file:

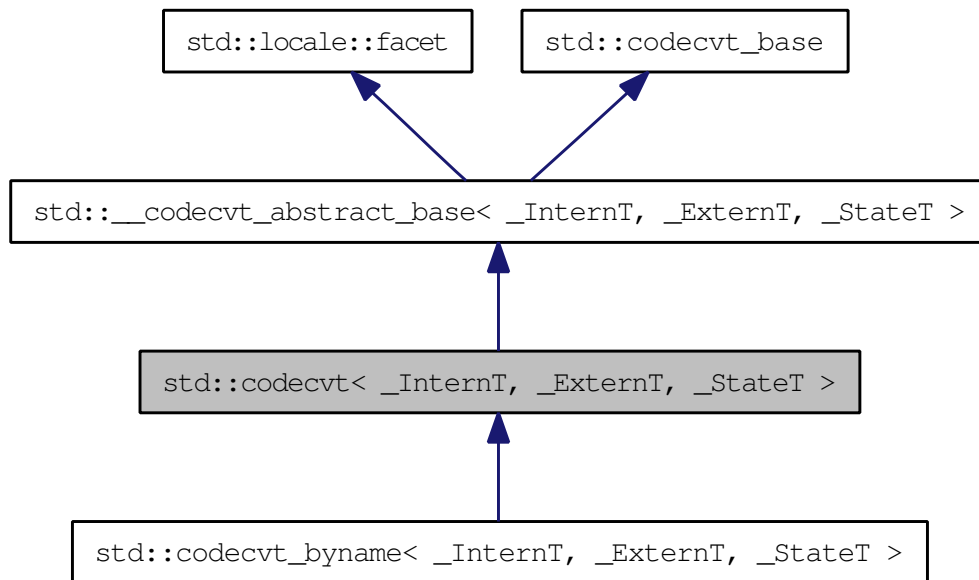
- [chrono](#)



## 5.412 `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` (`__c_locale __cloc`, `size_t __refs=0`)
- `codecvt` (`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

- **\_\_attribute\_\_** ((\_\_const\_\_)) static const char \*\_S\_get\_c\_name() throw ()
- 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 \_\_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`

### 5.412.1 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 275 of file `codecvt.h`.

### 5.412.2 Member Function Documentation

**5.412.2.1** `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 >`.

```
5.412.2.2 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 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

## 5.412 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference 2405

**5.412.2.3** `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 115 of file `codecvt.h`.

```
5.412.2.4 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 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.413 `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` (`state_type` &\_\_enc, `size_t` \_\_refs=0)
- `codecvt` (`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

- `__attribute__((__const__)) static const char *_S_get_c_name() throw ()`
- 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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.413.1 Detailed Description

`template<typename _InternT, typename _ExternT> class std::codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Definition at line 226 of file `codecvt_specializations.h`.



## 5.413.2 Member Function Documentation

**5.413.2.1** `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 [inline, 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 302 of file `codecvt_specializations.h`.

**5.413.2.2** `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 195 of file codecvt.h.

**5.413.2.3** `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 *& __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 115 of file `codecvt.h`.

**5.413.2.4** `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 154 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.414 `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` (`__c_locale __cloc`, `size_t __refs=0`)
- `codecvt` (`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

- `__attribute__((const))` static const char \*\_S\_get\_c\_name() throw ()
- 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 `__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`

### 5.414.1 Detailed Description

`template<> class std::codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

Definition at line 337 of file `codecvt.h`.

## 5.414.2 Member Function Documentation

**5.414.2.1** 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 >](#).

**5.414.2.2** 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 195 of file codecvt.h.

**5.414.2.3 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 115 of file `codecvt.h`.

**5.414.2.4** `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.

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 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecv.t.h](#)

## 5.415 `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` (`__c_locale __cloc`, `size_t __refs=0`)
- `codecvt` (`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

- `__attribute__((const))` static const char \*\_S\_get\_c\_name() throw ()
- 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 `__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`

### 5.415.1 Detailed Description

`template<> class std::codecvt< wchar_t, char, mbstate_t >`

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Definition at line 395 of file `codecvt.h`.

## 5.415.2 Member Function Documentation

**5.415.2.1** 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 >](#).

**5.415.2.2** 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 195 of file codecvt.h.

**5.415.2.3 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 115 of file `codecvt.h`.

**5.415.2.4** `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 154 of file `codecvt.h`.

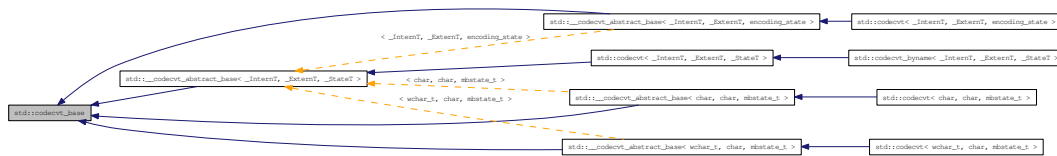
The documentation for this class was generated from the following file:

- [codecv.t.h](#)



## 5.416 std::codecvt\_base Class Reference

Empty base class for `codecvt` facet [22.2.1.5]. Inheritance diagram for `std::codecvt_base` - base:



### Public Types

- enum `result` { `ok`, `partial`, `error`, `noconv` }

### 5.416.1 Detailed Description

Empty base class for `codecvt` facet [22.2.1.5].

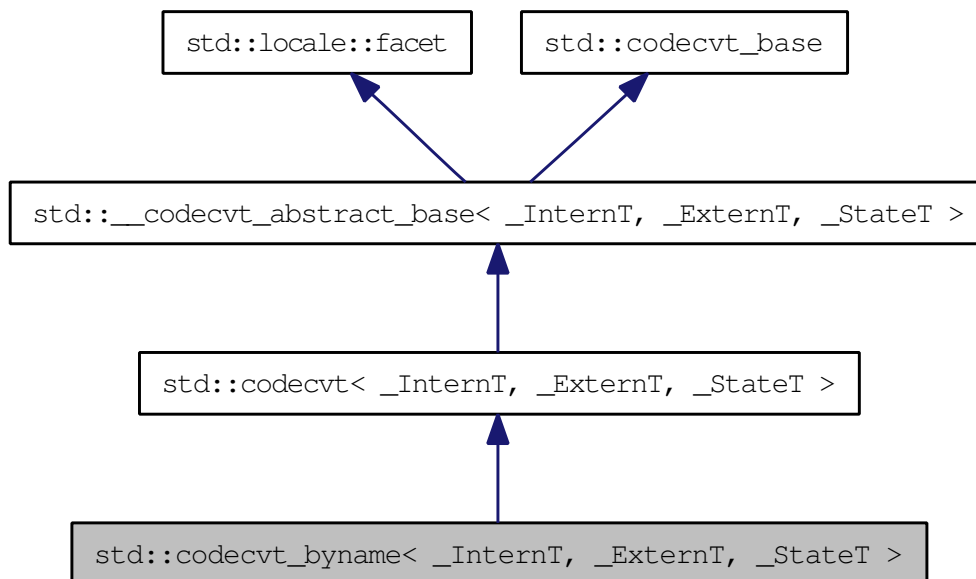
Definition at line 45 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.417 `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

- `__attribute__ ((__const__))` static `const char *_S_get_c_name` () `throw ()`
- 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 `__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_byname< _InternT, _ExternT, _StateT >
```

class `codecvt_byname` [22.2.1.6].

Definition at line 455 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`, `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 >`.

**5.417.2.2** `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.

## 5.417 `std::codecvt_byname< _InternT, _ExternT, _StateT >` Class Template Reference 2429

---

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 195 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 115 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.

## 5.417 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference 2431

---

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`.

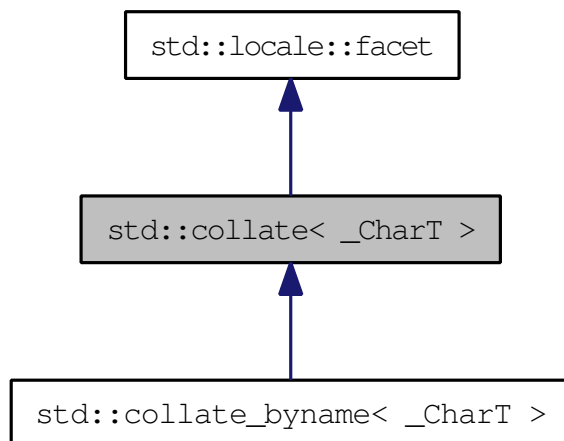
Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.418 `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` (`_c_locale __cloc`, `size_t __refs=0`)
- `collate` (`size_t __refs=0`)
- `template<>`  
`int _M_compare` (`const wchar_t *`, `const wchar_t *`) `const throw()`
- `template<>`  
`int _M_compare` (`const char *`, `const char *`) `const throw()`
- `int _M_compare` (`const _CharT *`, `const _CharT *`) `const throw ()`
- `template<>`  
`size_t _M_transform` (`wchar_t *`, `const wchar_t *`, `size_t`) `const throw()`
- `template<>`  
`size_t _M_transform` (`char *`, `const char *`, `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` ()
- `__attribute__` ((\_\_const\_\_)) static const char \*`_S_get_c_name`() throw ()
- 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 `__c_locale _S_lc_type_c_locale` (`__c_locale` `__cloc`, const char \*`__s`)

### Protected Attributes

- `__c_locale _M_c_locale_collate`

### Friends

- class `locale::_Impl`

#### 5.418.1 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 610 of file `locale_classes.h`.

## 5.418.2 Member Typedef Documentation

### 5.418.2.1 `template<typename _CharT> typedef _CharT std::collate< _CharT >::char_type`

Public typedefs.

Reimplemented in `std::collate_byname< _CharT >`.

Definition at line 616 of file `locale_classes.h`.

### 5.418.2.2 `template<typename _CharT> typedef basic_string< _CharT > std::collate< _CharT >::string_type`

Public typedefs.

Reimplemented in `std::collate_byname< _CharT >`.

Definition at line 617 of file `locale_classes.h`.

## 5.418.3 Constructor & Destructor Documentation

### 5.418.3.1 `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 637 of file `locale_classes.h`.

### 5.418.3.2 `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:**

- oloc* The C locale.
- refs* Passed to the base facet class.

Definition at line 651 of file locale\_classes.h.

**5.418.3.3** `template<typename _CharT> virtual std::collate<_CharT>::~~collate() [inline, protected, virtual]`

Destructor.

Definition at line 714 of file locale\_classes.h.

**5.418.4 Member Function Documentation**

**5.418.4.1** `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 668 of file locale\_classes.h.

**5.418.4.2** `template<typename _CharT> int std::collate<_CharT>::do_compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline, 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 134 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()`.

**5.418.4.3** `template<typename _CharT> long std::collate<_CharT>::do_hash(const _CharT * __lo, const _CharT * __hi) const [inline, 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 229 of file locale\_classes.tcc.

**5.418.4.4** `template<typename _CharT> collate<_CharT>::string_type std::collate<_CharT>::do_transform(const _CharT * __lo, const _CharT * __hi) const [inline, 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 173 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()`.

**5.418.4.5** `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 701 of file locale\_classes.h.

**5.418.4.6** `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 687 of file locale\_classes.h.

Referenced by std::regex\_traits< \_Ch\_type >::transform().

## 5.418.5 Member Data Documentation

### 5.418.5.1 `template<typename _CharT> locale::id std::collate< _CharT >::id` `[inline, static]`

Numpunct facet id.

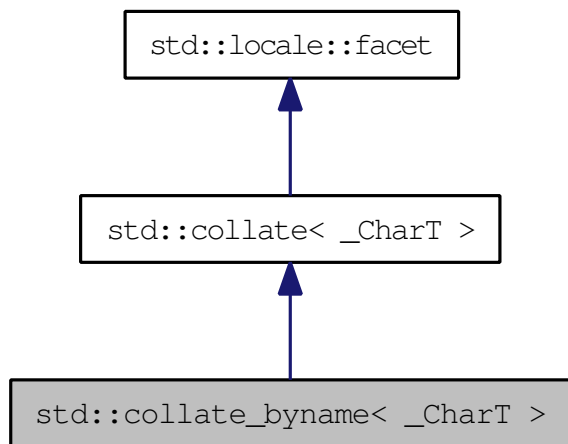
Definition at line 627 of file locale\_classes.h.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 5.419 `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`)
- `template<>`  
**`_M_compare`** (`const wchar_t*`, `const wchar_t*`) `const throw()`
- `template<>`  
**`_M_compare`** (`const char*`, `const char*`) `const throw()`
- **`_M_compare`** (`const _CharT*`, `const _CharT*`) `const throw ()`
- `template<>`  
**`_M_transform`** (`wchar_t*`, `const wchar_t*`, `size_t`) `const throw()`
- `template<>`  
**`_M_transform`** (`char*`, `const char*`, `size_t`) `const throw()`
- **`_M_transform`** (`_CharT*`, `const _CharT*`, `size_t`) `const throw ()`
- **`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

- [\\_\\_attribute\\_\\_](#) ((\_\_const\_\_)) static const char \*\_S\_get\_c\_name() throw ()
- 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 \_\_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](#)

### 5.419.1 Detailed Description

`template<typename _CharT> class std::collate_byname< _CharT >`

class [collate\\_byname](#) [22.2.4.2].

Definition at line 786 of file `locale_classes.h`.



## 5.419.2 Member Typedef Documentation

### 5.419.2.1 `template<typename _CharT > typedef _CharT std::collate_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::collate< \\_CharT >](#).

Definition at line 791 of file locale\_classes.h.

### 5.419.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::collate< \\_CharT >](#).

Definition at line 792 of file locale\_classes.h.

## 5.419.3 Member Function Documentation

### 5.419.3.1 `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 668 of file locale\_classes.h.

```
5.419.3.2 template<typename _CharT> int std::collate< _CharT
>::do_compare (const _CharT * __lo1, const _CharT * __hi1,
const _CharT * __lo2, const _CharT * __hi2) const [inline,
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 134 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()`.

```
5.419.3.3 template<typename _CharT> long std::collate< _CharT >::do_hash
(const _CharT * __lo, const _CharT * __hi) const [inline,
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 229 of file locale\_classes.tcc.

**5.419.3.4** `template<typename _CharT> collate< _CharT >::string_type  
std::collate< _CharT >::do_transform(const _CharT * __lo, const  
_CharT * __hi) const [inline, 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 173 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()`.

**5.419.3.5** `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 701 of file locale\_classes.h.

**5.419.3.6** `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 687 of file `locale_classes.h`.

Referenced by `std::regex_traits< _Ch_type >::transform()`.

## 5.419.4 Member Data Documentation

**5.419.4.1** `template<typename _CharT> locale::id std::collate< _CharT >::id`  
`[inline, static, inherited]`

Numpunct facet id.

Definition at line 627 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.420 `std::complex< _Tp >` Struct Template Reference

### Public Types

- typedef `_Tp` `value_type`

### Public Member Functions

- `template<typename _Up >`  
`complex` (const `complex< _Up >` &\_\_z)
- `complex` (const `_Tp` &\_\_r=`_Tp`(), const `_Tp` &\_\_i=`_Tp`())
- const `complex` & `__rep` () const
- void `imag` (`_Tp` \_\_val)
- `_Tp` `imag` () const
- `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` &\_\_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 >` &)
- `complex< _Tp >` & `operator/=` (const `_Tp` &)
- `template<typename _Up >`  
`complex< _Tp >` & `operator=` (const `complex< _Up >` &)
- `complex< _Tp >` & `operator=` (const `_Tp` &)
- void `real` (`_Tp` \_\_val)
- `_Tp` `real` () const

### 5.420.1 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 122 of file `complex`.

## 5.420.2 Member Typedef Documentation

### 5.420.2.1 `template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type`

Value typedef.

Definition at line 125 of file `complex`.

## 5.420.3 Constructor & Destructor Documentation

### 5.420.3.1 `template<typename _Tp> 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 129 of file `complex`.

### 5.420.3.2 `template<typename _Tp> template<typename _Up > std::complex< _Tp >::complex (const complex< _Up > & __z) [inline]`

Copy constructor.

Definition at line 136 of file `complex`.

## 5.420.4 Member Function Documentation

### 5.420.4.1 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+=( const _Tp & __t) [inline]`

Add `t` to this `complex` number.

Definition at line 179 of file `complex`.

### 5.420.4.2 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator-= (const _Tp & __t) [inline]`

Subtract `t` from this `complex` number.

Definition at line 188 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

## 5.421 `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 `_Clock`, typename `_Duration`, typename `_Predicate` >  
bool `wait_until` ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime, `_Predicate` \_\_p)
- template<typename `_Clock`, typename `_Duration` >  
`cv_status` `wait_until` ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime)
- template<typename `_Duration` >  
`cv_status` `wait_until` ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< `__clock_t`, `_Duration` > &\_\_atime)

### 5.421.1 Detailed Description

[condition\\_variable](#)

Definition at line 57 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)



## 5.422 `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` > &\_\_rtime, `_Predicate` \_\_p)
- template<typename `_Lock` , typename `_Rep` , typename `_Period` >  
[cv\\_status](#) **wait\_for** (`_Lock` &\_\_lock, const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rtime)
- template<typename `_Lock` , typename `_Clock` , typename `_Duration` , typename `_Predicate` >  
bool **wait\_until** (`_Lock` &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime, `_Predicate` \_\_p)
- template<typename `_Lock` , typename `_Clock` , typename `_Duration` >  
[cv\\_status](#) **wait\_until** (`_Lock` &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime)

### 5.422.1 Detailed Description

[condition\\_variable\\_any](#)

Definition at line 166 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.423 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

[conditional](#)

### Public Types

- `typedef _Iftrue type`

#### 5.423.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct
std::conditional< _Cond, _Iftrue, _Iffalse >
```

[conditional](#)

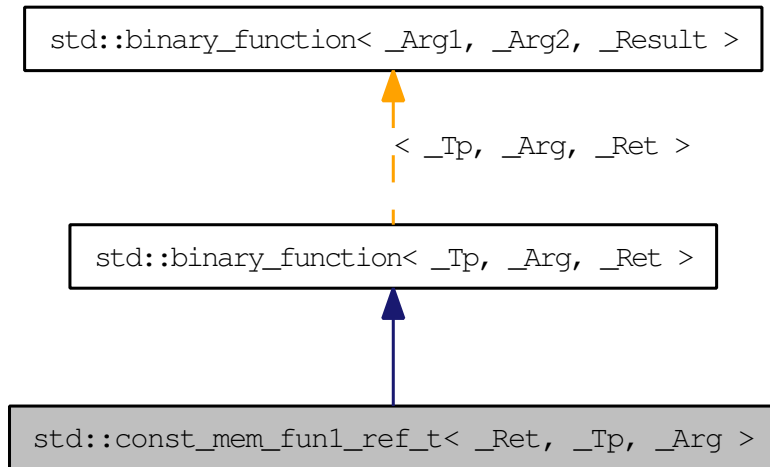
Definition at line 369 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.424 `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`

#### 5.424.1 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 650 of file `stl_function.h`.

## 5.424.2 Member Typedef Documentation

**5.424.2.1** `typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.424.2.2** `typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.424.2.3** `typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type [inherited]`

the type of the second argument

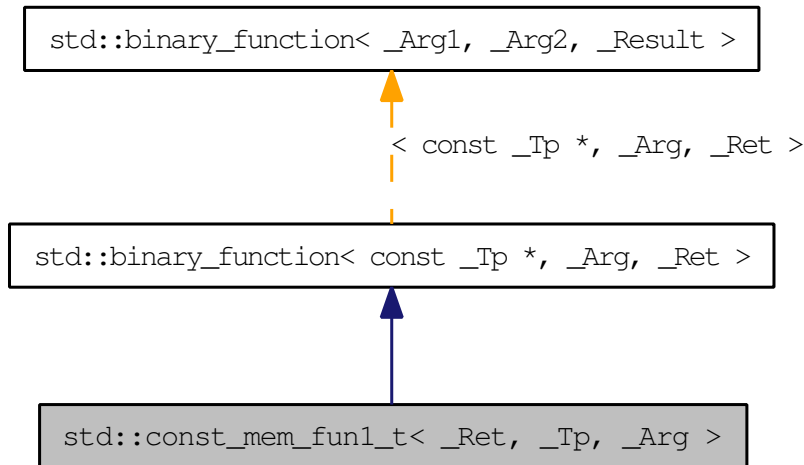
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.425 `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`

#### 5.425.1 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 614 of file `stl_function.h`.

## 5.425.2 Member Typedef Documentation

**5.425.2.1** `typedef const_Tp * std::binary_function< const_Tp *, _Arg , _Ret >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.425.2.2** `typedef _Ret std::binary_function< const_Tp *, _Arg , _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.425.2.3** `typedef _Arg std::binary_function< const_Tp *, _Arg , _Ret >::second_argument_type [inherited]`

the type of the second argument

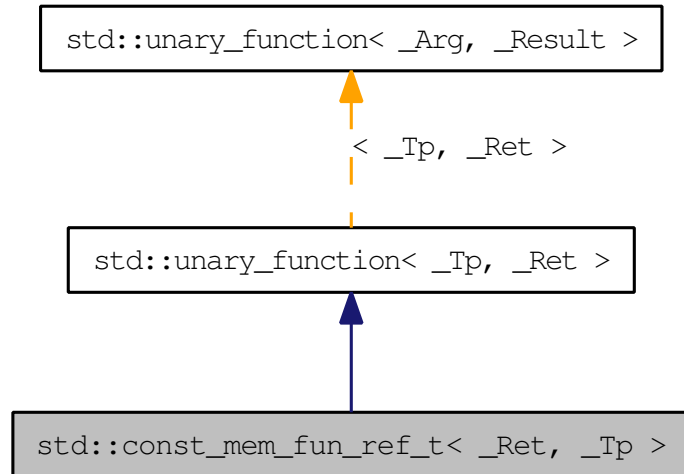
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.426 `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

#### 5.426.1 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 578 of file `stl_function.h`.

## 5.426.2 Member Typedef Documentation

### 5.426.2.1 `typedef _Tp std::unary_function< _Tp , _Ret >::argument_type` [`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.426.2.2 `typedef _Ret std::unary_function< _Tp , _Ret >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

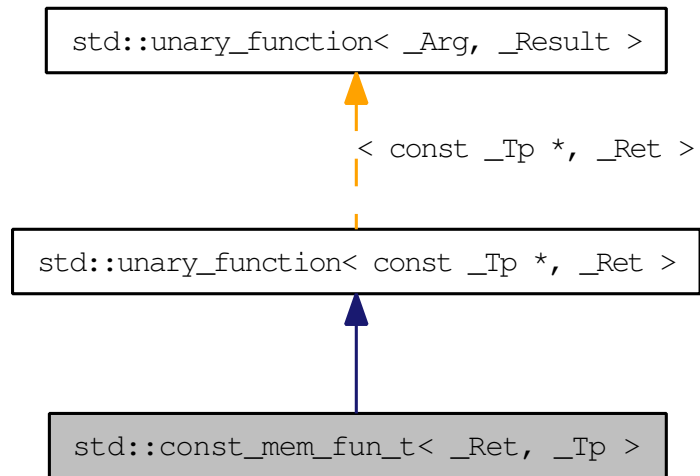
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)



## 5.427 `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`

#### 5.427.1 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 542 of file `stl_function.h`.

## 5.427.2 Member Typedef Documentation

### 5.427.2.1 `typedef const _Tp * std::unary_function< const _Tp * , _Ret >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.427.2.2 `typedef _Ret std::unary_function< const _Tp * , _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

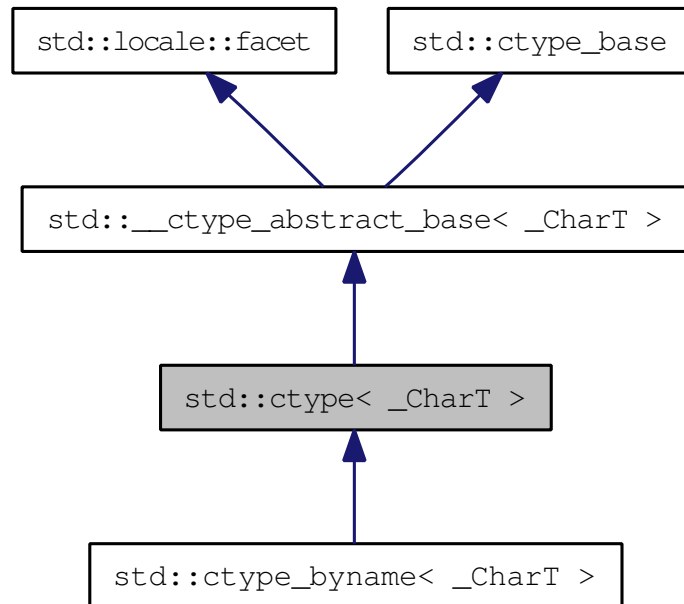
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.428 std::ctype< \_CharT > Class Template Reference

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. 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)
- const `char_type` \* **is** (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, `char_type` \_\_c) 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
- 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 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

- `__attribute__((__const__))` static const char \* `_S_get_c_name()` 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 const char\_type \* `do_tolower` (char\_type \*\_\_lo, const char\_type \*\_\_hi) const

- virtual `char_type do_tolower (char_type __c) 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 `__c_locale _S_lc_type_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

### 5.428.1 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 `cctype` 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 604 of file `locale_facets.h`.

## 5.428.2 Member Typedef Documentation

### 5.428.2.1 `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 608 of file locale\_facets.h.

## 5.428.3 Member Function Documentation

### 5.428.3.1 `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 >](#).

### 5.428.3.2 `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) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.428.3.3** `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>](#).

**5.428.3.4** `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>](#).

**5.428.3.5** `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>](#).

**5.428.3.6** `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>](#).

```
5.428.3.7 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>](#).

```
5.428.3.8 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>](#).

**5.428.3.9** `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 >`.

**5.428.3.10** `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 >`.

**5.428.3.11** `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 >](#).

**5.428.3.12** `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 >](#).

**5.428.3.13** `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::is(const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, 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 178 of file locale\_facets.h.

**5.428.3.14** `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:**

$(M \& m) \neq 0$ .

Definition at line 161 of file locale\_facets.h.

Referenced by `std::regex_traits<_Ch_type >::isctype()`, and `std::basic_istream<_CharT, _Traits >::sentry::sentry()`.

**5.428.3.15** `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, 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 345 of file locale\_facets.h.

**5.428.3.16** `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 323 of file locale\_facets.h.

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**5.428.3.17** `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 194 of file `locale_facets.h`.

**5.428.3.18** `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 210 of file `locale_facets.h`.

**5.428.3.19** `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_tolower(lo, hi)`.

**Parameters:**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns:**

*hi*.

Definition at line 268 of file `locale_facets.h`.

---

**5.428.3.20** `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_tolower(c)`.

**Parameters:**

*c* The `char_type` to convert.

**Returns:**

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

**5.428.3.21** `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 239 of file `locale_facets.h`.

**5.428.3.22** `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 224 of file locale\_facets.h.

**5.428.3.23** `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 [codecv](#) 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 304 of file locale\_facets.h.

**5.428.3.24** `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 [codecv](#) for that.

**Parameters:**

*c* The char to convert.



**Returns:**

The converted char\_type.

Definition at line 285 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<<().

**5.428.4 Member Data Documentation****5.428.4.1 template<typename \_CharT> locale::id std::ctype< \_CharT >::id  
[inline, static]**

The facet id for ctype<char\_type>.

Definition at line 612 of file locale\_facets.h.

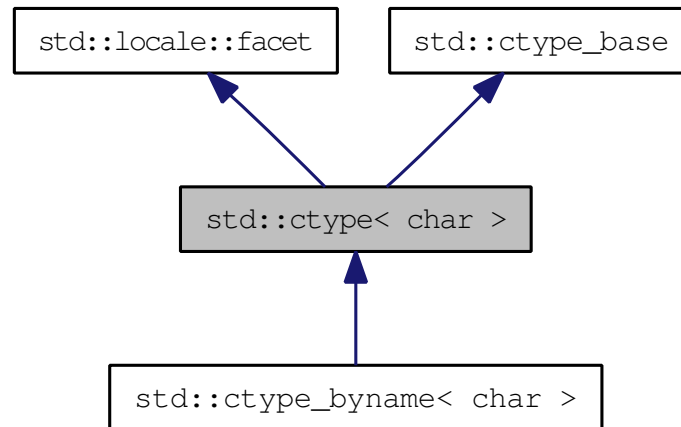
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.429 `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` (`__c_locale __cloc`, const mask \* `__table=0`, bool `__del=false`, size\_t `__refs=0`)
- `ctype` (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
- 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 \* `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
- 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 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` ()
- `__attribute__((__const__))` static const char \* `_S_get_c_name`() throw ()
- virtual const `char_type` \* `do_narrow` (const `char_type` \*\_\_lo, const `char_type` \*\_\_hi, char, char \*\_\_dest) const
- virtual char `do_narrow` (`char_type` \_\_c, char) const
- virtual const `char_type` \* `do_tolower` (`char_type` \*\_\_lo, const `char_type` \*\_\_hi) const
- virtual `char_type` `do_tolower` (`char_type`) 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 \_\_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 `__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`

### 5.429.1 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 673 of file `locale_facets.h`.

### 5.429.2 Member Typedef Documentation

#### 5.429.2.1 typedef `char std::ctype< char >::char_type`

Typedef for the template parameter char.

Definition at line 678 of file `locale_facets.h`.

### 5.429.3 Constructor & Destructor Documentation

**5.429.3.1** `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.

**5.429.3.2** `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.

**5.429.3.3** `virtual std::ctype< char >::~~ctype () [protected, virtual]`

Destructor. This function deletes `table()` if *del* was true in the constructor.

### 5.429.4 Member Function Documentation

**5.429.4.1** `static const mask* std::ctype< char >::classic_table () throw () [static]`

Returns a pointer to the C `locale` mask table.

**5.429.4.2 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 [codecv](#) 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 1149 of file locale\_facets.h.

**5.429.4.3 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 [codecv](#) for that.

**Parameters:**

- c* The char to convert.
- dfault* Char to return if conversion fails.

**Returns:**

The converted char.

Definition at line 1123 of file locale\_facets.h.

**5.429.4.4** `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*.

**5.429.4.5** `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*.

**5.429.4.6** `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*.

**5.429.4.7 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*.

**5.429.4.8 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 [codecv](#) 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 1097 of file locale\_facets.h.



**5.429.4.9 virtual char\_type std::ctype< char >::do\_widen (char \_\_c) const [inline, protected, virtual]**

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 1074 of file locale\_facets.h.

**5.429.4.10 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 46 of file ctype\_inline.h.

**5.429.4.11 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 41 of file `ctype_inline.h`.

**5.429.4.12** `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 955 of file `locale_facets.h`.

**5.429.4.13** `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 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 922 of file locale\_facets.h.

**5.429.4.14** `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 55 of file ctype\_inline.h.

**5.429.4.15** `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 65 of file ctype\_inline.h.

**5.429.4.16** `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 973 of file locale\_facets.h.

**5.429.4.17** `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 844 of file locale\_facets.h.

**5.429.4.18** `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 827 of file locale\_facets.h.

---

**5.429.4.19** `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 811 of file locale\_facets.h.

**5.429.4.20** `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 794 of file locale\_facets.h.

**5.429.4.21** `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 891 of file `locale_facets.h`.

**5.429.4.22 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 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 864 of file `locale_facets.h`.

**5.429.5 Member Data Documentation****5.429.5.1 locale::id std::ctype< char >::id [static]**

The facet id for `ctype<char>`.

Definition at line 695 of file `locale_facets.h`.

**5.429.5.2** `const size_t std::ctype< char >::table_size` [`static`]

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file `locale_facets.h`.

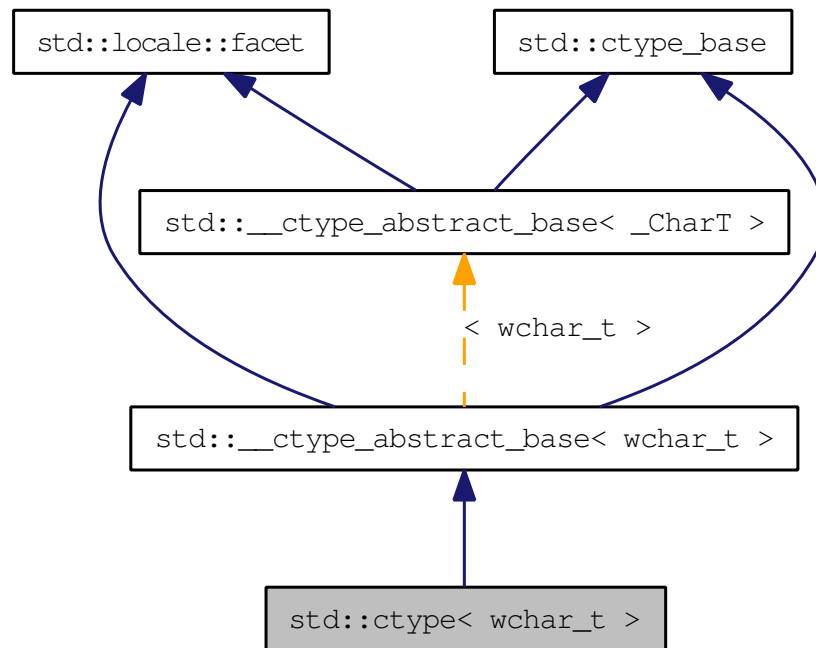
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

## 5.430 `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` (`__c_locale __cloc`, `size_t __refs=0`)
- `ctype` (`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
- 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
- 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 `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` ()
- `__attribute__((__const__)) static const char *_S_get_c_name` () throw ()
- `__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 __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 const `char_type * do_tolower` (`char_type * __lo`, const `char_type * __hi`) const
- virtual `char_type do_tolower` (`char_type`) 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 `__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`

### 5.430.1 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 1174 of file `locale_facets.h`.

## 5.430.2 Member Typedef Documentation

### 5.430.2.1 `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 1179 of file `locale_facets.h`.

## 5.430.3 Constructor & Destructor Documentation

### 5.430.3.1 `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.

### 5.430.3.2 `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.

### 5.430.3.3 `virtual std::ctype< wchar_t >::~~ctype () [protected, virtual]`

Destructor.

## 5.430.4 Member Function Documentation

**5.430.4.1** `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 >`.

**5.430.4.2** `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:

$(M \& m) \neq 0$ .

Implements `std::__ctype_abstract_base< wchar_t >`.

**5.430.4.3 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 undervived [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 >`.

**5.430.4.4 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 undervived [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 >`.

**5.430.4.5** `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 >](#).

**5.430.4.6** `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 >](#).

**5.430.4.7** `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 >`.

**5.430.4.8** `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 >`.

**5.430.4.9** `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 >](#).

**5.430.4.10 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 >](#).

**5.430.4.11 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 >](#).

**5.430.4.12 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 >](#).

**5.430.4.13 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 178 of file locale\_facets.h.

**5.430.4.14** `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:**

$(M \& m) \neq 0$ .

Definition at line 161 of file locale\_facets.h.

**5.430.4.15** `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 `codecv`t 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 345 of file locale\_facets.h.

---

**5.430.4.16** 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 323 of file locale\_facets.h.

**5.430.4.17** 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 194 of file locale\_facets.h.

**5.430.4.18** 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 210 of file locale\_facets.h.

**5.430.4.19** `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_tolower(lo, hi)`.

**Parameters:**

- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns:**

*hi*.

Definition at line 268 of file locale\_facets.h.

**5.430.4.20** `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_tolower(c)`.

**Parameters:**

- c* The `char_type` to convert.

**Returns:**

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file locale\_facets.h.

---

**5.430.4.21** `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 239 of file locale\_facets.h.

**5.430.4.22** `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().

**Parameters:**

*c* The char\_type to convert.

**Returns:**

The uppercase char\_type if convertible, else *c*.

Definition at line 224 of file locale\_facets.h.

**5.430.4.23** `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 304 of file locale\_facets.h.

**5.430.4.24 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 285 of file locale\_facets.h.

**5.430.5 Member Data Documentation****5.430.5.1 locale::id std::ctype< wchar\_t >::id [static]**

The facet id for [ctype<wchar\\_t>](#).

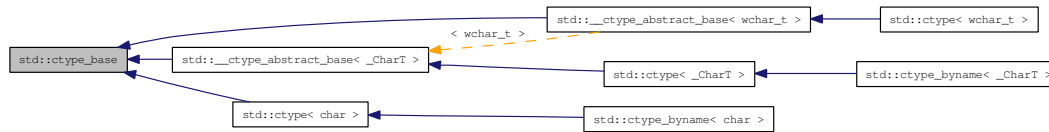
Definition at line 1197 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.431 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**

#### 5.431.1 Detailed Description

Base class for [ctype](#).

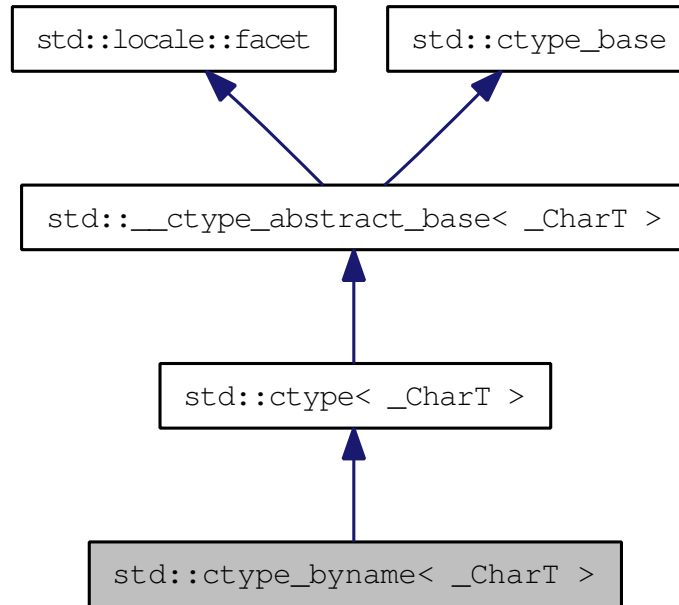
Definition at line 40 of file [ctype\\_base.h](#).

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 5.432 `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`)
- `const char_type * is` (`const char_type * __lo, const char_type * __hi, mask * __vec`) `const`
- `bool is` (`mask __m, char_type __c`) `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
- 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 `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

- `__attribute__` ((\_\_const\_\_)) static const `char` \* `_S_get_c_name`() 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` `__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 const `char_type` \* `do_tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- virtual `char_type` `do_tolower` (`char_type` `__c`) 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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

### 5.432.1 Detailed Description

`template<typename _CharT> class std::ctype_byname< _CharT >`

class `ctype_byname` [22.2.1.2].

Definition at line 1466 of file `locale_facets.h`.

### 5.432.2 Member Typedef Documentation

**5.432.2.1** `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 608 of file `locale_facets.h`.

### 5.432.3 Member Function Documentation

**5.432.3.1** `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>](#).

**5.432.3.2** `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) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.432.3.3** `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 >](#).

**5.432.3.4** `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 >](#).

```
5.432.3.5 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 >](#).

```
5.432.3.6 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 >](#).

**5.432.3.7** `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 >`.

**5.432.3.8** `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 >`.

**5.432.3.9** `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>`.

**5.432.3.10** `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>`.

**5.432.3.11** `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>](#).

**5.432.3.12** `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 [codecv](#) for that.

**Parameters:**

*c* The char to convert.

**Returns:**

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.432.3.13** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT >::is(const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, 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 178 of file locale\_facets.h.

**5.432.3.14** `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:**

$(M \& m) \neq 0$ .

Definition at line 161 of file locale\_facets.h.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.432.3.15** `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, 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 345 of file `locale_facets.h`.

**5.432.3.16** `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 323 of file `locale_facets.h`.

Referenced by `std::time_put<_CharT, _OutIter >::put()`.

**5.432.3.17** `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 194 of file locale\_facets.h.

```
5.432.3.18 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 210 of file locale\_facets.h.

```
5.432.3.19 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\\_tolower\(lo, hi\)](#).

**Parameters:**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns:**

*hi*.

Definition at line 268 of file locale\_facets.h.

**5.432.3.20** `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_tolower(c)`.

**Parameters:**

*c* The `char_type` to convert.

**Returns:**

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

**5.432.3.21** `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 239 of file `locale_facets.h`.

**5.432.3.22** `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 224 of file locale\_facets.h.

**5.432.3.23** `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 304 of file locale\_facets.h.

**5.432.3.24** `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 285 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<<().

### 5.432.4 Member Data Documentation

#### 5.432.4.1 `template<typename _CharT> locale::id std::ctype<_CharT >::id` `[inline, static, inherited]`

The facet id for ctype<char\_type>.

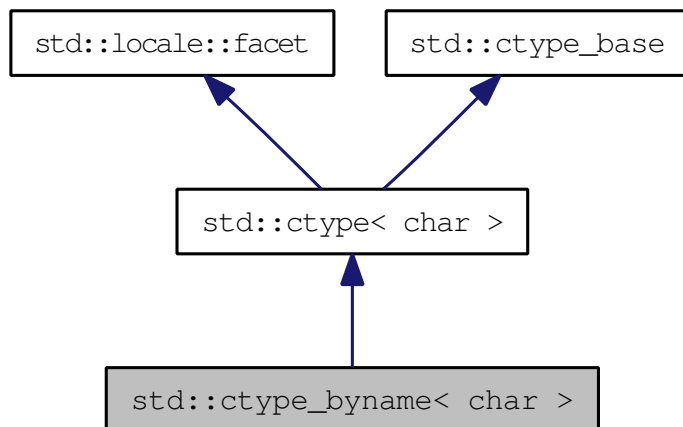
Definition at line 612 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.433 `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`)
- const char \* `is` (const char \* `__lo`, const char \* `__hi`, mask \* `__vec`) const
- bool `is` (mask `__m`, char `__c`) 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 \* `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
- 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 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

- [\\_\\_attribute\\_\\_](#) ((\_\_const\_\_)) static const char \* [\\_S\\_get\\_c\\_name](#)() throw ()
- virtual const [char\\_type \\* do\\_narrow](#) (const [char\\_type \\* \\_\\_lo](#), const [char\\_type \\* \\_\\_hi](#), char, char \* \_\_dest) const
- virtual char [do\\_narrow](#) ([char\\_type \\_\\_c](#), char) const
- virtual const [char\\_type \\* do\\_tolower](#) ([char\\_type \\* \\_\\_lo](#), const [char\\_type \\* \\_\\_hi](#)) const
- virtual [char\\_type do\\_tolower](#) ([char\\_type](#)) 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 \_\_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 `__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`

### 5.433.1 Detailed Description

`template<> class std::ctype_byname< char >`

22.2.1.4 Class [ctype\\_byname](#) specializations.

Definition at line 1481 of file `locale_facets.h`.

### 5.433.2 Member Typedef Documentation

**5.433.2.1** `typedef char std::ctype< char >::char_type` `[inherited]`

Typedef for the template parameter `char`.

Definition at line 678 of file `locale_facets.h`.

### 5.433.3 Member Function Documentation

**5.433.3.1** `static const mask* std::ctype< char >::classic_table () throw ()`  
**[static, inherited]**

Returns a pointer to the C [locale](#) mask table.

**5.433.3.2** `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 [codecv](#) 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 1149 of file [locale\\_facets.h](#).

**5.433.3.3** `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 [codecv](#) for that.

**Parameters:**

- c* The char to convert.  
*dfault* Char to return if conversion fails.

**Returns:**

The converted char.

Definition at line 1123 of file locale\_facets.h.

**5.433.3.4 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*.

**5.433.3.5 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*.

**5.433.3.6** `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*.

**5.433.3.7** `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*.

**5.433.3.8** `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 [codecv](#) 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 1097 of file `locale_facets.h`.

**5.433.3.9** `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 1074 of file `locale_facets.h`.

**5.433.3.10** `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 46 of file ctype\_inline.h.

**5.433.3.11 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 41 of file ctype\_inline.h.

**5.433.3.12 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 955 of file locale\_facets.h.

**5.433.3.13 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 922 of file locale\_facets.h.

**5.433.3.14 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 55 of file ctype\_inline.h.

**5.433.3.15 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 65 of file ctype\_inline.h.

#### 5.433.3.16 `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 973 of file locale\_facets.h.

#### 5.433.3.17 `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 844 of file locale\_facets.h.

#### 5.433.3.18 `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 827 of file `locale_facets.h`.

**5.433.3.19** `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 811 of file `locale_facets.h`.

**5.433.3.20** `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 794 of file locale\_facets.h.

**5.433.3.21** `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 [codecv](#) 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 891 of file locale\_facets.h.

**5.433.3.22** `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 [codecv](#) for that.

**Parameters:**

- c* The char to convert.

**Returns:**

The converted character.

Definition at line 864 of file locale\_facets.h.

---

## 5.433.4 Member Data Documentation

### 5.433.4.1 locale::id std::ctype< char >::id [static, inherited]

The facet id for [ctype<char>](#).

Definition at line 695 of file locale\_facets.h.

### 5.433.4.2 const size\_t std::ctype< char >::table\_size [static, inherited]

The size of the mask table. It is SCHAR\_MAX + 1.

Definition at line 697 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.434 `std::decay< _Tp >` Class Template Reference

[decay](#)

### Public Types

- `typedef __decay_selector< __remove_type >::__type type`

### 5.434.1 Detailed Description

`template<typename _Tp> class std::decay< _Tp >`

[decay](#)

Definition at line 400 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 5.435 std::decimal::decimal128 Class Reference

3.2.4 Class [decimal128](#).

### Public Types

- typedef float \_\_decfloat128 **\_\_attribute\_\_** ((mode(TD)))

### Public Member Functions

- [decimal128](#) (\_\_decfloat128 \_\_z)
- **decimal128** (unsigned long long \_\_z)
- **decimal128** (long long \_\_z)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (long \_\_z)
- **decimal128** (unsigned int \_\_z)
- **decimal128** (int \_\_z)
- **decimal128** (long double \_\_r)
- **decimal128** (double \_\_r)
- **decimal128** (float \_\_r)
- **decimal128** ([decimal64](#) d64)
- **decimal128** ([decimal32](#) d32)
- \_\_decfloat128 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat128 \_\_x)
- [decimal128](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator\*=** (long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator\*=** (long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator\*=** (int \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator++** ()
- [decimal128](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator+=** (long long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator+=** (long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator+=** (int \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) \_\_rhs)

- [decimal128](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator--** ()
- [decimal128](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator-=** (long long \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator-=** (long \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator-=** (int \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)

### 5.435.1 Detailed Description

3.2.4 Class [decimal128](#).

Definition at line 391 of file decimal.

### 5.435.2 Constructor & Destructor Documentation

#### 5.435.2.1 `std::decimal::decimal128::decimal128 (__decfloat128 __z)` [`inline`]

Conforming extension: Conversion from scalar [decimal](#) type.

Definition at line 416 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.436 std::decimal::decimal32 Class Reference

3.2.2 Class [decimal32](#).

### Public Types

- typedef float \_\_decfloat32 **\_\_attribute\_\_** ((mode(SD)))

### Public Member Functions

- [decimal32](#) (\_\_decfloat32 \_\_z)
- **decimal32** (unsigned long long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (int \_\_z)
- **decimal32** (long double \_\_r)
- **decimal32** (double \_\_r)
- **decimal32** (float \_\_r)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** ([decimal64](#) \_\_d64)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- [decimal32](#) & **operator\***=(unsigned long long \_\_rhs)
- [decimal32](#) & **operator\***=(long long \_\_rhs)
- [decimal32](#) & **operator\***=(unsigned long \_\_rhs)
- [decimal32](#) & **operator\***=(long \_\_rhs)
- [decimal32](#) & **operator\***=(unsigned int \_\_rhs)
- [decimal32](#) & **operator\***=(int \_\_rhs)
- [decimal32](#) & **operator\***=([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\***=([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\***=([decimal32](#) \_\_rhs)
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator++** ()
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (int \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)

- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator--** ()
- [decimal32](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator-=** (long long \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator-=** (long \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)

### 5.436.1 Detailed Description

3.2.2 Class [decimal32](#).

Definition at line 225 of file decimal.

### 5.436.2 Constructor & Destructor Documentation

#### 5.436.2.1 `std::decimal::decimal32::decimal32 (__decfloat32 __z) [inline]`

Conforming extension: Conversion from scalar [decimal](#) type.

Definition at line 249 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)



## 5.437 std::decimal::decimal64 Class Reference

3.2.3 Class [decimal64](#).

### Public Types

- typedef float \_\_decfloat64 **\_\_attribute\_\_** ((mode(DD)))

### Public Member Functions

- [decimal64](#) (\_\_decfloat64 \_\_z)
- **decimal64** (unsigned long long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (int \_\_z)
- **decimal64** (long double \_\_r)
- **decimal64** (double \_\_r)
- **decimal64** (float \_\_r)
- **decimal64** ([decimal128](#) d128)
- **decimal64** ([decimal32](#) d32)
- \_\_decfloat64 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- [decimal64](#) & **operator\***=(unsigned long long \_\_rhs)
- [decimal64](#) & **operator\***=(long long \_\_rhs)
- [decimal64](#) & **operator\***=(unsigned long \_\_rhs)
- [decimal64](#) & **operator\***=(long \_\_rhs)
- [decimal64](#) & **operator\***=(unsigned int \_\_rhs)
- [decimal64](#) & **operator\***=(int \_\_rhs)
- [decimal64](#) & **operator\***=([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator\***=([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator\***=([decimal32](#) \_\_rhs)
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator++** ()
- [decimal64](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator+=** (long long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator+=** (long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator+=** (int \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) \_\_rhs)

- [decimal64](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator--** ()
- [decimal64](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator-=** (long long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator-=** (long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator-=** (int \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator/=** (long long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator/=** (long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator/=** (int \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal32](#) \_\_rhs)

### 5.437.1 Detailed Description

3.2.3 Class [decimal64](#).

Definition at line 308 of file decimal.

### 5.437.2 Constructor & Destructor Documentation

#### 5.437.2.1 `std::decimal::decimal64::decimal64(__decfloat64 __z) [inline]`

Conforming extension: Conversion from scalar [decimal](#) type.

Definition at line 332 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.438 `std::default_delete< _Tp >` Struct Template Reference

Primary template, [default\\_delete](#).

### Public Member Functions

- `template<typename _Up > default_delete` (const [default\\_delete](#)< \_Up > &)
- `void operator()` (\_Tp \* \_\_ptr) const

### 5.438.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp >`

Primary template, [default\\_delete](#).

Definition at line 48 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.439 `std::default_delete< _Tp[]>` Struct Template Reference

Specialization, [default\\_delete](#).

### Public Member Functions

- `void operator() (_Tp *__ptr) const`

#### 5.439.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp[]>`

Specialization, [default\\_delete](#).

Definition at line 68 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.440 `std::defer_lock_t` Struct Reference

Do not acquire ownership of the [mutex](#).

### 5.440.1 Detailed Description

Do not acquire ownership of the [mutex](#).

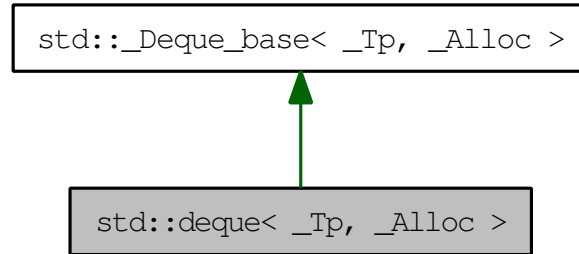
Definition at line 376 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.441 `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

- `template<typename _InputIterator >`  
`deque` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const allocator_type &__a=allocator_type()`)
- `deque` (`initializer_list< value_type >` \_\_l, `const allocator_type &__a=allocator_type()`)
- `deque` (`deque &&__x`)
- `deque` (`const deque &__x`)
- `deque` (`size_type __n`, `const value_type &__value=value_type()`, `const allocator_type &__a=allocator_type()`)

- [deque](#) (const allocator\_type &\_\_a)
- [deque](#) ()
- [~deque](#) ()
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- 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)
- template<typename... \_Args >  
void [emplace\\_back](#) (\_Args &&... \_\_args)
- template<typename... \_Args >  
void [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const
- [const\\_iterator end](#) () const
- [iterator end](#) ()
- [iterator erase](#) (iterator \_\_first, iterator \_\_last)
- [iterator erase](#) (iterator \_\_position)
- const\_reference [front](#) () const
- reference [front](#) ()
- allocator\_type [get\\_allocator](#) () const
- template<typename \_InputIterator >  
void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- void [insert](#) (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- [iterator insert](#) (iterator \_\_position, 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=](#) (deque &&\_\_x)
- [deque & operator=](#) (const deque &\_\_x)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const

- reference [operator\[\]](#) (size\_type \_\_n)
- 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)
- [const\\_reverse\\_iterator rbegin](#) () const
- [reverse\\_iterator rbegin](#) ()
- [const\\_reverse\\_iterator rend](#) () const
- [reverse\\_iterator rend](#) ()
- void [resize](#) (size\_type \_\_new\_size, value\_type \_\_x=value\_type())
- 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 \\_InputIterator >](#)  
void [\\_M\\_assign\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- [template<typename \\_Integer >](#)  
void [\\_M\\_assign\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_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\\_destroy\\_data](#) ([iterator](#) \_\_first, [iterator](#) \_\_last, const [std::allocator< \\_Tp >](#) &)



- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `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`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `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)`
- `void _M_initialize_map (size_t)`
- `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... _Args >`  
`iterator _M_insert_aux (iterator __pos, _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 __x, __true_`  
`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)`
  
- `void _M_new_elements_at_back (size_type __new_elements)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
- `template<typename... _Args >`  
`void _M_push_back_aux (_Args &&... __args)`

- `template<typename... _Args>`  
`void \_M\_push\_front\_aux (_Args &&...__args)`
- `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)`
- `void \_M\_reallocate\_map (size_type __nodes_to_add, bool __add_at_front)`
- `iterator \_M\_reserve\_elements\_at\_back (size_type __n)`
- `iterator \_M\_reserve\_elements\_at\_front (size_type __n)`
- `void \_M\_reserve\_map\_at\_back (size_type __nodes_to_add=1)`
- `void \_M\_reserve\_map\_at\_front (size_type __nodes_to_add=1)`

## Static Protected Member Functions

- `static size_t \_S\_buffer\_size ()`

## Protected Attributes

- `\_Deque\_impl \_M\_impl`

### 5.441.1 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 713 of file `stl_deque.h`.

## 5.441.2 Constructor & Destructor Documentation

**5.441.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::deque () [inline]`

Default constructor creates no elements.

Definition at line 765 of file `stl_deque.h`.

**5.441.2.2** `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 773 of file `stl_deque.h`.

**5.441.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::deque (size_type __n, const value_type  
& __value = value_type (), const allocator_type & __a =  
allocator_type ()) [inline, explicit]`

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 785 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_fill_initialize()`.

**5.441.2.4** `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 797 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

**5.441.2.5** `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 811 of file `stl_deque.h`.

**5.441.2.6** `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 825 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_initialize()`.

```
5.441.2.7 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 logN memory reallocations.

Definition at line 850 of file `stl_deque.h`.

```
5.441.2.8 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 864 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

### 5.441.3 Member Function Documentation

```
5.441.3.1 template<typename _Tp , typename _Alloc > void
 deque::_M_fill_initialize (const value_type & __value) [inline,
 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 277 of file deque.tcc.

References `std::_Destroy()`.

Referenced by `std::deque< _Tp, _Alloc >::deque()`.

**5.441.3.2** `template<typename _Tp , typename _Alloc > void  
std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t  
__num_elements) [inline, 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 568 of file stl\_deque.h.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_range_initialize()`.

**5.441.3.3** `template<typename _Tp , typename _Alloc > void  
deque::_M_new_elements_at_back (size_type __new_elements)  
[inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 771 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()`.

**5.441.3.4** `template<typename _Tp, typename _Alloc > void deque::_M_new_elements_at_front (size_type __new_elements) [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 746 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()`.

**5.441.3.5** `template<typename _Tp, typename _Alloc > void deque::_M_pop_back_aux () [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 427 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_back()`.

**5.441.3.6** `template<typename _Tp, typename _Alloc > void deque::_M_pop_front_aux () [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 442 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_front()`.

**5.441.3.7** `template<typename _Tp, typename _Alloc > template<typename... _Args> void deque::_M_push_back_aux (_Args &&... __args) [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 361 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::push_back()`.



**5.441.3.8** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> void deque::_M_push_front_aux (_Args &&... _args)  
[inline, protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 395 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_front().

**5.441.3.9** `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 1159 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::at().

**5.441.3.10** `template<typename _Tp, typename _Alloc > template<typename  
_ForwardIterator > void deque::_M_range_initialize  
(_ForwardIterator __first, _ForwardIterator __last,  
std::forward_iterator_tag) [inline, protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 323 of file deque.tcc.

References std::\_Destroy(), std::\_Deque\_base< \_Tp, \_Alloc >::\_M\_initialize\_map(),  
std::advance(), and std::distance().

**5.441.3.11** `template<typename _Tp, typename _Alloc > template<typename  
_InputIterator > void deque::_M_range_initialize (_InputIterator  
__first, _InputIterator __last, std::input_iterator_tag) [inline,  
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 303 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()`.

**5.441.3.12** `template<typename _Tp, typename _Alloc > void  
deque::_M_reallocate_map (size_type __nodes_to_add, bool  
__add_at_front) [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 796 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()`.

**5.441.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc >::_M_reserve_elements_at_back  
(size_type __n) [inline, protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 1768 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_new_elements_at_back()`.

**5.441.3.14** `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 1758 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_new_elements_at_front()`.

**5.441.3.15** `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 1794 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**5.441.3.16** `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]`

Helper functions for `push_*` and `pop_*`.

Definition at line 1802 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**5.441.3.17** `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 961 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::assign()`.

Referenced by `std::deque< _Tp, _Alloc >::assign()`.

```
5.441.3.18 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 942 of file stl\_deque.h.

```
5.441.3.19 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 925 of file stl\_deque.h.

Referenced by `std::deque< _Tp, _Alloc >::operator=()`.

```
5.441.3.20 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 1196 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_range_check()`.

**5.441.3.21** `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 1178 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_range_check()`.

**5.441.3.22** `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 1235 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::end()`.

**5.441.3.23** `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 1223 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::end()`.

**5.441.3.24** `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 984 of file `stl_deque.h`.

**5.441.3.25** `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 976 of file `stl_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::operator==(,)`, and `std::deque<_Tp, _Alloc >::~~deque()`.

**5.441.3.26** `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 1047 of file `stl_deque.h`.

**5.441.3.27** `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 1056 of file `stl_deque.h`.

**5.441.3.28** `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 1501 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=()`.

**5.441.3.29** `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 1065 of file stl\_deque.h.

**5.441.3.30** `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 1074 of file stl\_deque.h.

**5.441.3.31** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> deque< _Tp, _Alloc >::iterator deque::emplace (iterator  
_position, _Args &&... _args) [inline]`

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 144 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()`.

**5.441.3.32** `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 1122 of file `stl_deque.h`.

**5.441.3.33** `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 1002 of file `stl_deque.h`.

**5.441.3.34** `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 993 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc >::back()`, `std::deque<_Tp, _Alloc >::deque()`, `std::deque<_Tp, _Alloc >::erase()`, `std::operator==(, and std::deque<_Tp, _Alloc >::~~deque().`

**5.441.3.35** `template<typename _Tp, typename _Alloc > deque<_Tp,`  
`_Alloc >::iterator deque::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 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 189 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\(\)](#).

**5.441.3.36** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator __position) [inline]`

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 166 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\(\)](#).

**5.441.3.37** `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 1215 of file stl\_deque.h.

References [std::deque< \\_Tp, \\_Alloc >::begin\(\)](#).

**5.441.3.38** `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 1207 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::begin()`.

**5.441.3.39** `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 967 of file `stl_deque.h`.

**5.441.3.40** `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 1428 of file `stl_deque.h`.

**5.441.3.41** `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 1413 of file stl\_deque.h.

```
5.441.3.42 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 1399 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::insert().

Referenced by std::deque< \_Tp, \_Alloc >::insert().

```
5.441.3.43 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 1386 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::emplace().

**5.441.3.44** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::insert (iterator __position, const value_type & __x) [inline]`

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 121 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::push_back()`, and `std::deque< _Tp, _Alloc >::push_front()`.

Referenced by `std::deque< _Tp, _Alloc >::resize()`.

**5.441.3.45** `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 1086 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()`.

**5.441.3.46** `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 907 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::assign().

**5.441.3.47** `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 886 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::clear(), and std::deque< \_Tp, \_Alloc >::swap().

**5.441.3.48** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc  
> & deque::operator= (const deque< _Tp, _Alloc > & _x)  
[inline]`

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 65 of file deque.tcc.

**5.441.3.49** `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 1153 of file `stl_deque.h`.

**5.441.3.50** `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 1138 of file `stl_deque.h`.

**5.441.3.51** `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 1336 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_back_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::erase()`.

**5.441.3.52** `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 1315 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_pop_front_aux()`.

Referenced by `std::deque< _Tp, _Alloc >::erase()`.

**5.441.3.53** `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 1284 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()`.

**5.441.3.54** `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 1253 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()`.

**5.441.3.55** `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 1020 of file `stl_deque.h`.

**5.441.3.56** `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 1011 of file `stl_deque.h`.

**5.441.3.57** `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 1038 of file `stl_deque.h`.

**5.441.3.58** `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 1029 of file `stl_deque.h`.

**5.441.3.59** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::resize(size_type __new_size,  
value_type __x = value_type()) [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 1101 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::insert()`, and `std::deque< _Tp, _Alloc >::size()`.

**5.441.3.60** `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 1113 of file stl\_deque.h.

**5.441.3.61** `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 1081 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::operator==(,)`, and `std::deque< _Tp, _Alloc >::resize()`.

**5.441.3.62** `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 1481 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](#)

## 5.442 `std::discard_block_engine<_RandomNumberEngine, __p, __r >` Class Template Reference

### Public Types

- typedef `_RandomNumberEngine::result_type` [result\\_type](#)

### Public Member Functions

- `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](#) (`result_type __s`)
- [discard\\_block\\_engine](#) (`_RandomNumberEngine &&__rne`)
- [discard\\_block\\_engine](#) (`const _RandomNumberEngine &__rne`)
- [discard\\_block\\_engine](#) ()
- `const _RandomNumberEngine &` [base](#) () `const`
- `void` [discard](#) (`unsigned long long __z`)
- `result_type` [max](#) () `const`
- `result_type` [min](#) () `const`
- `result_type` [operator\(\)](#) ()
- `template<typename _Sseq >`  
`void` [seed](#) (`_Sseq &__q`)
- `void` [seed](#) (`result_type __s`)
- `void` [seed](#) ()

### Static Public Attributes

- static const `size_t` [block\\_size](#)
- static const `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 > &)`

### 5.442.1 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.

$0 \leq \text{__r} \leq \text{__p}$

Definition at line 716 of file random.h.

### 5.442.2 Member Typedef Documentation

5.442.2.1 `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 723 of file random.h.

### 5.442.3 Constructor & Destructor Documentation

5.442.3.1 `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 734 of file random.h.

**5.442 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference** **2573**

---

**5.442.3.2** `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:**

*rne* An existing (base class) engine object.

Definition at line 744 of file random.h.

**5.442.3.3** `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:**

*rne* An existing (base class) engine object.

Definition at line 754 of file random.h.

**5.442.3.4** `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 764 of file random.h.

```
5.442.3.5 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 777 of file random.h.

## 5.442.4 Member Function Documentation

```
5.442.4.1 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 821 of file random.h.

```
5.442.4.2 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.

**Todo**

Look for a faster way to do discard.

Definition at line 848 of file random.h.

```
5.442.4.3 template<typename _RandomNumberEngine, size_t __p, size_t __r>
result_type std::discard_block_engine< _RandomNumberEngine,
__p, __r >::max () const [inline]
```

Gets the maximum value in the generated random number range.

**5.442 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference** **2575**

---

**Todo**

This should be constexpr.

Definition at line 839 of file random.h.

**5.442.4.4** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
result_type std::discard_block_engine<_RandomNumberEngine,  
__p, __r >::min () const [inline]`

Gets the minimum value in the generated random number range.

**Todo**

This should be constexpr.

Definition at line 830 of file random.h.

**5.442.4.5** `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() () [inline]`

Gets the next value in the generated random number sequence.

Definition at line 652 of file random.tcc.

**5.442.4.6** `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 810 of file random.h.

**5.442.4.7** `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 797 of file random.h.

**5.442.4.8** `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 786 of file random.h.

## 5.442.5 Friends And Related Function Documentation

**5.442.5.1** `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.

**5.442.5.2** `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.



**5.442 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference** **2577**

---

*\_\_rhs* Another discard\_block\_engine random number generator object.

**Returns:**

true if the two objects are equal, false otherwise.

Definition at line 871 of file random.h.

**5.442.5.3** `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](#)

## 5.443 `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

- **discrete\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_Func` >  
**discrete\_distribution** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, `_Func` \_\_fw)
- **discrete\_distribution** ([initializer\\_list](#)< double > \_\_wl)
- template<typename `_InputIterator` >  
**discrete\_distribution** (`_InputIterator` \_\_wbegin, `_InputIterator` \_\_wend)
- [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` > &)

### 5.443.1 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 4089 of file `random.h`.

### 5.443.2 Member Typedef Documentation

**5.443.2.1** `template<typename _IntType = int> typedef _IntType  
std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4096 of file `random.h`.

### 5.443.3 Member Function Documentation

**5.443.3.1** `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 4198 of file `random.h`.

References `std::vector< _Tp, _Alloc >::size()`.

**5.443.3.2** `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 4191 of file `random.h`.

**5.443.3.3** `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 4184 of file random.h.

**5.443.3.4** `template<typename _IntType = int> param_type  
std::discrete_distribution< _IntType >::param () const [inline]`

Returns the parameter `set` of the distribution.

Definition at line 4176 of file random.h.

**5.443.3.5** `template<typename _IntType = int> std::vector<double>  
std::discrete_distribution< _IntType >::probabilities () const  
[inline]`

Returns the probabilities of the distribution.

Definition at line 4169 of file random.h.

**5.443.3.6** `template<typename _IntType = int> void std::discrete_distribution<  
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 4162 of file random.h.

## 5.443.4 Friends And Related Function Documentation

**5.443.4.1** `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.

**5.443.4.2** `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.

`__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](#)

## 5.444 `std::discrete_distribution<_IntType>::param_type` Struct Reference

### Public Types

- typedef [discrete\\_distribution](#)<\_IntType > `distribution_type`

### Public Member Functions

- `template<typename _Func > param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)`
- `param_type (initializer_list< double > __wil)`
- `template<typename _InputIterator > param_type (_InputIterator __wbegin, _InputIterator __wend)`
- `std::vector< double > probabilities () const`

### Friends

- class `discrete_distribution<_IntType >`

#### 5.444.1 Detailed Description

`template<typename _IntType = int> struct std::discrete_distribution<_IntType>::param_type`

Parameter type.

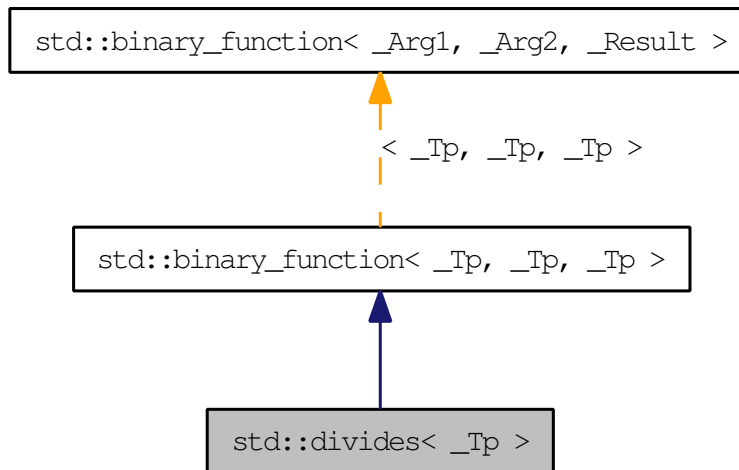
Definition at line 4098 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.445 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`

#### 5.445.1 Detailed Description

```
template<typename _Tp> struct std::divides< _Tp >
```

One of the [math functors](#).

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

## 5.445.2 Member Typedef Documentation

**5.445.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::first_argument_type` [*inherited*]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.445.2.2** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type`  
[*inherited*]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.445.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::second_argument_type` [*inherited*]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

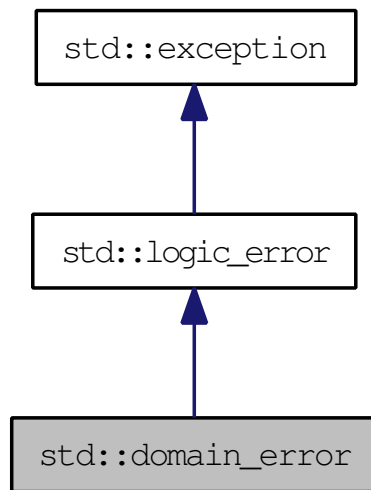
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



## 5.446 `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 ()

#### 5.446.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 73 of file `stdexcept`.

#### 5.446.2 Member Function Documentation

##### 5.446.2.1 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](#).

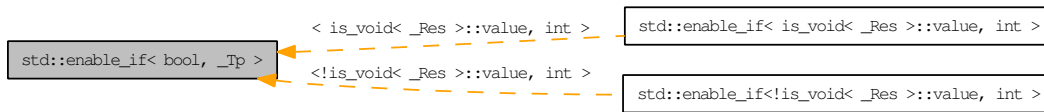
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.447 `std::enable_if< bool, _Tp >` Struct Template Reference

`enable_if` Inheritance diagram for `std::enable_if< bool, _Tp >`:



### 5.447.1 Detailed Description

```
template<bool, typename _Tp = void> struct std::enable_if< bool, _Tp >
```

`enable_if`

Definition at line 356 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.448 `std::enable_shared_from_this< _Tp >` Class Template Reference

Base class allowing use of member function `shared_from_this`.

### Public Member Functions

- [shared\\_ptr](#)< const \_Tp > `shared_from_this` () const
- [shared\\_ptr](#)< \_Tp > `shared_from_this` ()

### 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`)

### 5.448.1 Detailed Description

`template<typename _Tp> class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

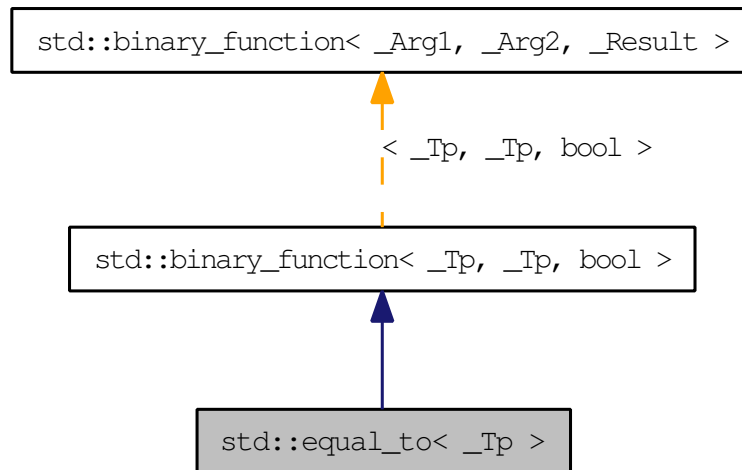
Definition at line 399 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.449 std::equal\_to< \_Tp > Struct Template Reference

One of the [comparison functors](#). 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`

#### 5.449.1 Detailed Description

```
template<typename _Tp> struct std::equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 199 of file `stl_function.h`.

## 5.449.2 Member Typedef Documentation

**5.449.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [*inherited*]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.449.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [*inherited*]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.449.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [*inherited*]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.450 `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

### 5.450.1 Detailed Description

[error\\_category](#)

Definition at line 64 of file `system_error`.

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.451 std::error\_code Struct Reference

[error\\_code](#)

### Public Member Functions

- `template<typename _ErrorCodeEnum >`  
`error_code` (`_ErrorCodeEnum __e`, `typename enable\_if< is\_error\_code\_enum< _ErrorCodeEnum >::value >::type * = 0`)
- `error_code` (`int __v`, `const error\_category & __cat`)
- `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 >`

#### 5.451.1 Detailed Description

[error\\_code](#)

Definition at line 116 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)



## 5.452 `std::error_condition` Struct Reference

[error\\_condition](#)

### Public Member Functions

- `template<typename _ErrorConditionEnum >`  
`error_condition` (`_ErrorConditionEnum __e`, `typename enable_if< is_error_`  
`condition_enum< _ErrorConditionEnum >::value >::type *=0`)
- `error_condition` (`int __v`, `const error_category &__cat`)
- `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`

### 5.452.1 Detailed Description

[error\\_condition](#)

Definition at line 193 of file `system_error`.

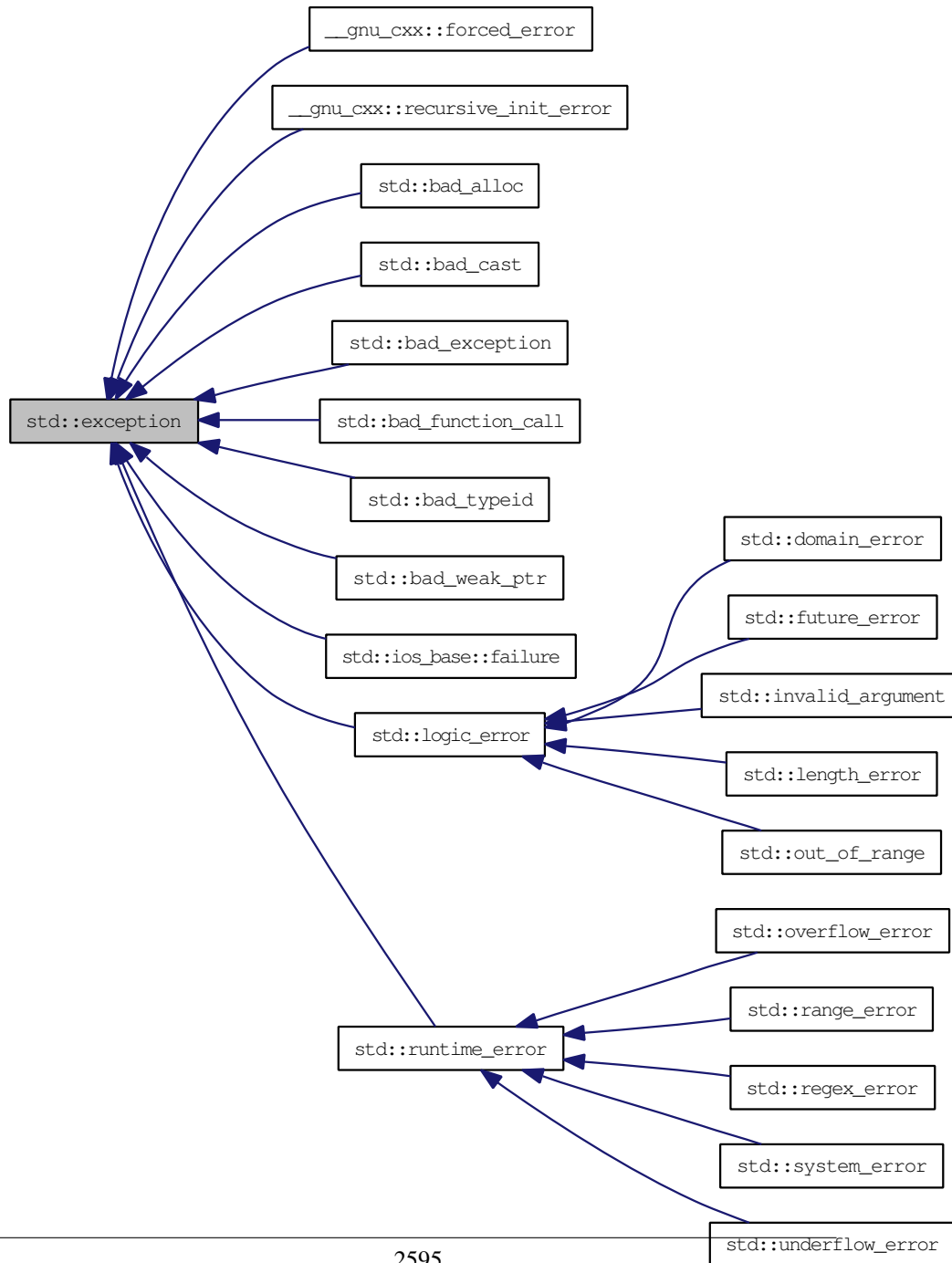
The documentation for this struct was generated from the following file:

- [system\\_error](#)



## 5.453 std::exception Class Reference

Base class for all library exceptions. Inheritance diagram for std::exception:



## Public Member Functions

- virtual const char \* [what](#) () const throw ()

### 5.453.1 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`.

### 5.453.2 Member Function Documentation

#### 5.453.2.1 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::bad\\_weak\\_ptr](#), and [std::ios\\_base::failure](#).

The documentation for this class was generated from the following file:

- [exception](#)

## **5.454 `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 [param\\_type](#) &\_\_p)
- **exponential\_distribution** (const [result\\_type](#) &\_\_lambda=[result\\_type](#)(1))
- `_RealType` **lambda** () 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** ()

### **5.454.1 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  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

Definition at line 3649 of file random.h.

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda}$     |

Table 5.1: Distribution Statistics

## 5.454.2 Member Typedef Documentation

**5.454.2.1** `template<typename _RealType = double> typedef _RealType  
std::exponential_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 3656 of file random.h.

## 5.454.3 Constructor & Destructor Documentation

**5.454.3.1** `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 3683 of file random.h.

## 5.454.4 Member Function Documentation

**5.454.4.1** `template<typename _RealType = double> _RealType  
std::exponential_distribution< _RealType >::lambda () const  
[inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 3704 of file random.h.

**5.454.4.2** `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 3733 of file random.h.

## 5.454 std::exponential\_distribution<\_RealType> Class Template Reference 2599

**5.454.4.3** `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 3726 of file random.h.

**5.454.4.4** `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 3719 of file random.h.

**5.454.4.5** `template<typename _RealType = double> param_type  
std::exponential_distribution<_RealType>::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 3711 of file random.h.

Referenced by `std::operator>>()`.

**5.454.4.6** `template<typename _RealType = double> void  
std::exponential_distribution<_RealType>::reset () [inline]`

Resets the distribution state. Has no effect on exponential distributions.

Definition at line 3698 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.455 `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

#### 5.455.1 Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution<_RealType>::param_type`

Parameter type.

Definition at line 3658 of file `random.h`.

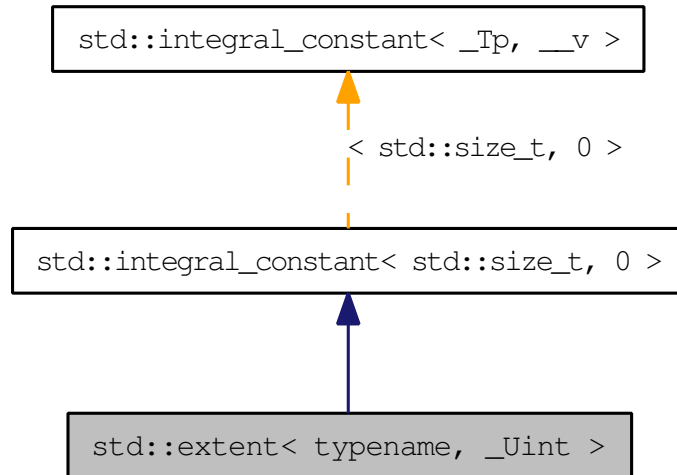
The documentation for this struct was generated from the following file:

- [random.h](#)



## 5.456 `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**

### Static Public Attributes

- static const `std::size_t` **value**

#### 5.456.1 Detailed Description

`template<typename, unsigned _Uint = 0> struct std::extent< typename, _Uint >`

[extent](#)

Definition at line 368 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.457 `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](#) (const [param\\_type](#) &\_\_p)
- [extreme\\_value\\_distribution](#) (`_RealType` \_\_a=`_RealType`(0), `_RealType` \_\_b=`_RealType`(1))
- `_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](#) ()

#### 5.457.1 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 3944 of file random.h.

## 5.457.2 Member Typedef Documentation

5.457.2.1 `template<typename _RealType = double> typedef _RealType  
std::extreme_value_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 3951 of file `random.h`.

## 5.457.3 Member Function Documentation

5.457.3.1 `template<typename _RealType = double> _RealType  
std::extreme_value_distribution<_RealType>::a () const  
[inline]`

Return the  $a$  parameter of the distribution.

Definition at line 3998 of file `random.h`.

5.457.3.2 `template<typename _RealType = double> _RealType  
std::extreme_value_distribution<_RealType>::b () const  
[inline]`

Return the  $b$  parameter of the distribution.

Definition at line 4005 of file `random.h`.

5.457.3.3 `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 4034 of file `random.h`.

5.457.3.4 `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 4027 of file `random.h`.

**5.457.3.5** `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 4020 of file random.h.

**5.457.3.6** `template<typename _RealType = double> param_type  
std::extreme_value_distribution< _RealType >::param () const  
[inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 4012 of file random.h.

Referenced by `std::operator>>()`.

**5.457.3.7** `template<typename _RealType = double> void  
std::extreme_value_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 3991 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.458 `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

### 5.458.1 Detailed Description

`template<typename _RealType = double> struct std::extreme_value_distribution<_RealType>::param_type`

Parameter type.

Definition at line 3953 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.459 `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** (const [param\\_type](#) &\_\_p)
- **fisher\_f\_distribution** (`_RealType` \_\_m=`_RealType`(1), `_RealType` \_\_n=`_RealType`(1))
- `_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` , typename `_CharT` , typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & **operator>>** (`std::basic_istream`< `_CharT`, `_Traits` > &, [std::fisher\\_f\\_distribution](#)< `_RealType1` > &)

### 5.459.1 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 2562 of file random.h.

### 5.459.2 Member Typedef Documentation

**5.459.2.1** `template<typename _RealType = double> typedef _RealType std::fisher_f_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2569 of file random.h.

### 5.459.3 Member Function Documentation

**5.459.3.1** `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 2652 of file random.h.

**5.459.3.2** `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 2645 of file random.h.

**5.459.3.3** `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 2638 of file random.h.

**5.459.3.4** `template<typename _RealType = double> param_type  
std::fisher_f_distribution<_RealType >::param () const [inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 2630 of file random.h.

**5.459.3.5** `template<typename _RealType = double> void  
std::fisher_f_distribution<_RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2609 of file random.h.

References `std::gamma_distribution<_RealType >::reset()`.

**5.459.4 Friends And Related Function Documentation**

**5.459.4.1** `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.



5.459.4.2 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::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](#)

## 5.460 `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

#### 5.460.1 Detailed Description

`template<typename _RealType = double> struct std::fisher_f_distribution<_RealType >::param_type`

Parameter type.

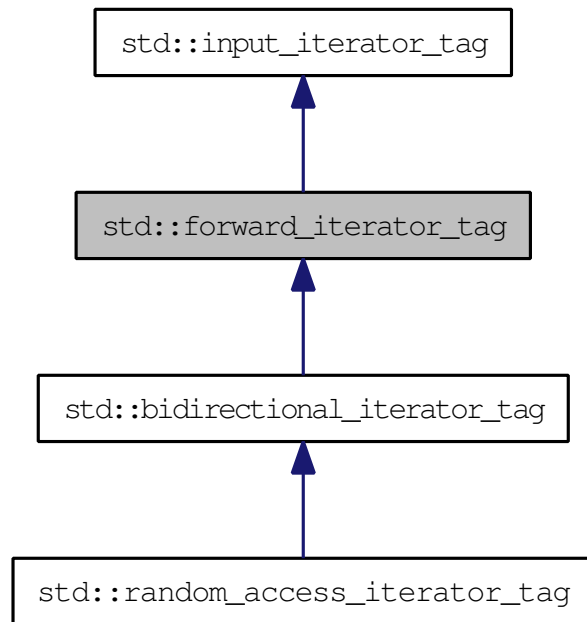
Definition at line 2571 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.461 `std::forward_iterator_tag` Struct Reference

Forward iterators support a superset of input [iterator](#) operations. Inheritance diagram for `std::forward_iterator_tag`:



### 5.461.1 Detailed Description

Forward iterators support a superset of input [iterator](#) operations.

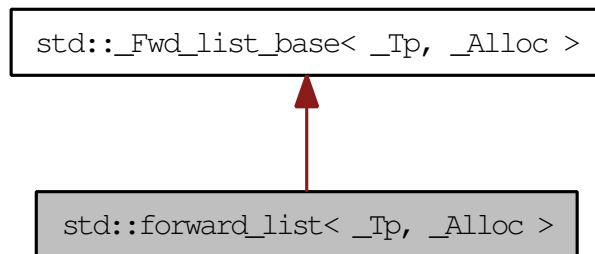
Definition at line 85 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.462 `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 `_Base::const_iterator` **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 `_Base::iterator` **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` (`std::initializer_list< _Tp > __il`, `const _Alloc &__al=_Alloc()`)
- `forward_list` (`forward_list &&__list`)
- `forward_list` (`const forward_list &__list`)
- `template<typename _InputIterator >`  
`forward_list` (`_InputIterator __first`, `_InputIterator __last`, `const _Alloc &__al=_Alloc()`)
- `forward_list` (`size_type __n`, `const _Tp &__value`, `const _Alloc &__al=_Alloc()`)
- `forward_list` (`size_type __n`)
- `forward_list` (`forward_list &&__list`, `const _Alloc &__al`)
- `forward_list` (`const forward_list &__list`, `const _Alloc &__al`)

- `forward_list` (const `_Alloc` &\_\_al=`_Alloc`())
- `~forward_list` ()
- void `assign` (`std::initializer_list`< `_Tp` > \_\_il)
- void `assign` (size\_type \_\_n, const `_Tp` &\_\_val)
- template<typename `_InputIterator` >  
void `assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `const_iterator before_begin` () const
- `iterator before_begin` ()
- `const_iterator begin` () const
- `iterator begin` ()
- `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` ()
- void `erase_after` (`const_iterator` \_\_pos, `const_iterator` \_\_last)
- void `erase_after` (`const_iterator` \_\_pos)
- const\_reference `front` () const
- reference `front` ()
- allocator\_type `get_allocator` () const
- `iterator insert_after` (`const_iterator` \_\_pos, `std::initializer_list`< `_Tp` > \_\_il)
- template<typename `_InputIterator` >  
`iterator insert_after` (`const_iterator` \_\_pos, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator insert_after` (`const_iterator` \_\_pos, size\_type \_\_n, const `_Tp` &\_\_val)
- `iterator insert_after` (`const_iterator` \_\_pos, `_Tp` &&\_\_val)
- `iterator insert_after` (`const_iterator` \_\_pos, const `_Tp` &\_\_val)
- 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=` (`forward_list` &&\_\_list)
- `forward_list` & `operator=` (const `forward_list` &\_\_list)
- void `pop_front` ()
- void `push_front` (`_Tp` &&\_\_val)
- void `push_front` (const `_Tp` &\_\_val)

- void `remove` (const `_Tp` &\_\_val)
- template<typename `_Pred` >  
void `remove_if` (`_Pred` \_\_pred)
- void `resize` (size\_type \_\_sz, value\_type \_\_val)
- void `resize` (size\_type \_\_sz)
- 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, _Tp_alloc_type >>::other _Node_alloc_type`

### Private Member Functions

- template<typename... `_Args`>  
`_Node::Pointer` `_M_create_node` (`_Args` &&...\_\_args)
- void `_M_erase_after` (typename `_Node_base::Pointer` \_\_pos, typename `_Node_base::Pointer` \_\_last)
- void `_M_erase_after` (typename `_Node_base::Pointer` \_\_pos)
- `_Node::Pointer` `_M_get_node` ()
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const
- `_Node_alloc_type` & `_M_get_Node_allocator` ()
- template<typename... `_Args`>  
`_Node_base::Pointer` `_M_insert_after` (const\_iterator \_\_pos, `_Args` &&...\_\_args)
- void `_M_put_node` (typename `_Node::Pointer` \_\_p)

### Private Attributes

- `_Fwd_list_impl` `_M_impl`

### 5.462.1 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 389 of file `forward_list.h`.

### 5.462.2 Constructor & Destructor Documentation

```
5.462.2.1 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 418 of file `forward_list.h`.

```
5.462.2.2 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 427 of file `forward_list.h`.

**5.462.2.3** `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 436 of file `forward_list.h`.

**5.462.2.4** `template<typename _Tp, typename _Alloc > 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 179 of file `forward_list.tcc`.

**5.462.2.5** `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 459 of file forward\_list.h.

```
5.462.2.6 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 475 of file forward\_list.h.

```
5.462.2.7 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 492 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::begin(), and std::forward\_list< \_Tp, \_-  
Alloc >::end().

```
5.462.2.8 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 505 of file `forward_list.h`.

**5.462.2.9** `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 516 of file `forward_list.h`.

**5.462.2.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::~~forward_list () [inline]`

The `forward_list` dtor.

Definition at line 524 of file `forward_list.h`.

### 5.462.3 Member Function Documentation

**5.462.3.1** `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 618 of file `forward_list.h`.

**5.462.3.2** `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 603 of file forward\_list.h.

**5.462.3.3** `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 586 of file forward\_list.h.

**5.462.3.4** `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 645 of file forward\_list.h.

**5.462.3.5** `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 636 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc >::operator=()`, and `std::forward_list<_Tp, _Alloc >::resize()`.

**5.462.3.6** `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 662 of file `forward_list.h`.

**5.462.3.7** `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 653 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()`.

**5.462.3.8** `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 698 of file `forward_list.h`.

**5.462.3.9** `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 689 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==(()).

**5.462.3.10** `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 707 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==(()).

**5.462.3.11** `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 1019 of file forward\_list.h.

**5.462.3.12** `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 822 of file `forward_list.h`.

**5.462.3.13** `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 766 of file `forward_list.h`.

**5.462.3.14** `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 715 of file `forward_list.h`.

**5.462.3.15** `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 680 of file `forward_list.h`.

**5.462.3.16** `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 671 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().

**5.462.3.17** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void 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.

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 960 of file forward\_list.h.

**5.462.3.18** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void 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.

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 936 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()`.

**5.462.3.19** `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 744 of file `forward_list.h`.

**5.462.3.20** `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 732 of file `forward_list.h`.

**5.462.3.21** `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 626 of file `forward_list.h`.

**5.462.3.22** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list<_Tp, _Alloc >::insert_after  
(const_iterator __pos, std::initializer_list<_Tp > __il) [inline]`

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:**

*pos*.

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 912 of file forward\_list.h.

```
5.462.3.23 template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator > iterator std::forward_list<
_Tp, _Alloc >::insert_after (const_iterator __pos, _InputIterator
__first, _InputIterator __last) [inline]
```

Inserts a range into the forward\_list.

**Parameters:**

*position* An iterator into the forward\_list.

*first* An input iterator.

*last* An input iterator.

**Returns:**

pos.

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 888 of file forward\_list.h.

```
5.462.3.24 template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after
(const_iterator __pos, size_type __n, const _Tp & __val)
[inline]
```

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:**

pos.

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 864 of file `forward_list.h`.

```
5.462.3.25 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 839 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::operator=()`, and `std::forward_list< _Tp, _Alloc >::resize()`.

```
5.462.3.26 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 722 of file `forward_list.h`.

```
5.462.3.27 template<typename _Tp , typename _Alloc > template<typename
_Comp > void std::forward_list< _Tp, _Alloc >::merge
(forward_list< _Tp, _Alloc > && __list, _Comp __comp)
[inline]
```

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 355 of file forward\_list.tcc.

```
5.462.3.28 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 1146 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::merge().

Referenced by std::forward\_list< \_Tp, \_Alloc >::merge().

```
5.462.3.29 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 566 of file forward\_list.h.

**5.462.3.30** `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 548 of file `forward_list.h`.

References `std::swap()`.

**5.462.3.31** `template<typename _Tp, typename _Alloc > forward_list<_Tp,  
_Alloc > & std::forward_list<_Tp, _Alloc >::operator= (const  
forward_list<_Tp, _Alloc > & __list) [inline]`

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 193 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()`.

**5.462.3.32** `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 804 of file `forward_list.h`.

**5.462.3.33** `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 781 of file `forward_list.h`.

**5.462.3.34** `template<typename _Tp , typename _Alloc > void  
std::forward_list<_Tp, _Alloc >::remove (const _Tp & __val)  
[inline]`

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 294 of file `forward_list.tcc`.

**5.462.3.35** `template<typename _Tp , typename _Alloc > template<typename  
_Pred > void std::forward_list<_Tp, _Alloc >::remove_if (_Pred  
__pred) [inline]`

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 314 of file `forward_list.tcc`.

**5.462.3.36** `template<typename _Tp, typename _Alloc > void  
std::forward_list<_Tp, _Alloc >::resize (size_type __sz, value_type  
__val) [inline]`

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 242 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()`.

**5.462.3.37** `template<typename _Tp, typename _Alloc > void  
std::forward_list<_Tp, _Alloc >::resize (size_type __sz)  
[inline]`

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 220 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 >::splice_after()`.

**5.462.3.38** `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 1190 of file `forward_list.h`.

**5.462.3.39** `template<typename _Tp, class _Alloc > template<typename  
_Comp > void std::forward_list< _Tp, _Alloc >::sort (_Comp  
_comp) [inline]`

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 401 of file `forward_list.tcc`.

**5.462.3.40** `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 1171 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::sort()`.

Referenced by `std::forward_list< _Tp, _Alloc >::sort()`.

**5.462.3.41** `template<typename _Tp, typename _Alloc > void  
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator __pos,  
forward_list< _Tp, _Alloc > && __list, const_iterator __before,  
const_iterator __last) [inline]`

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 278 of file `forward_list.tcc`.

**5.462.3.42** `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 1049 of file forward\_list.h.

**5.462.3.43** `template<typename _Tp, typename _Alloc > 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 261 of file forward\_list.tcc.

Referenced by std::forward\_list< \_Tp, \_Alloc >::resize().

**5.462.3.44** `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 978 of file forward\_list.h.

References std::swap().

Referenced by std::swap().



**5.462.3.45** `template<typename _Tp, typename _Alloc > template<typename _BinPred > void std::forward_list< _Tp, _Alloc >::unique (_BinPred __binary_pred) [inline]`

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 334 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()`.

**5.462.3.46** `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 1117 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::unique()`.

Referenced by `std::forward_list< _Tp, _Alloc >::unique()`.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.463 `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)
- [streamoff operator-](#) (const [fpos](#) &\_\_other) const
- [fpos operator-](#) ([streamoff](#) \_\_off) const
- [fpos & operator-=](#) ([streamoff](#) \_\_off)
- [\\_StateT state](#) () const
- void [state](#) ([\\_StateT](#) \_\_st)

### 5.463.1 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 112 of file `postypes.h`.

### 5.463.2 Constructor & Destructor Documentation

**5.463.2.1** `template<typename _StateT> std::fpos<_StateT >::fpos (streamoff __off) [inline]`

Construct position from offset.

Definition at line 133 of file `postypes.h`.

### 5.463.3 Member Function Documentation

**5.463.3.1** `template<typename _StateT> std::fpos< _StateT >::operator streamoff () const [inline]`

Convert to streamoff.

Definition at line 137 of file postypes.h.

**5.463.3.2** `template<typename _StateT> fpos std::fpos< _StateT >::operator+ (streamoff __off) const [inline]`

Add position and offset.

Definition at line 178 of file postypes.h.

**5.463.3.3** `template<typename _StateT> fpos& std::fpos< _StateT >::operator+= (streamoff __off) [inline]`

Add offset to this position.

Definition at line 154 of file postypes.h.

**5.463.3.4** `template<typename _StateT> streamoff std::fpos< _StateT >::operator- (const fpos< _StateT > & __other) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

**5.463.3.5** `template<typename _StateT> fpos std::fpos< _StateT >::operator- (streamoff __off) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

**5.463.3.6** `template<typename _StateT> fpos& std::fpos< _StateT >::operator-= (streamoff __off) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.

**5.463.3.7** `template<typename _StateT> _StateT std::fpos< _StateT >::state () const [inline]`

Return the last [set](#) value of *st*.

Definition at line 146 of file `postypes.h`.

**5.463.3.8** `template<typename _StateT> void std::fpos< _StateT >::state (_StateT __st) [inline]`

Remember the value of *st*.

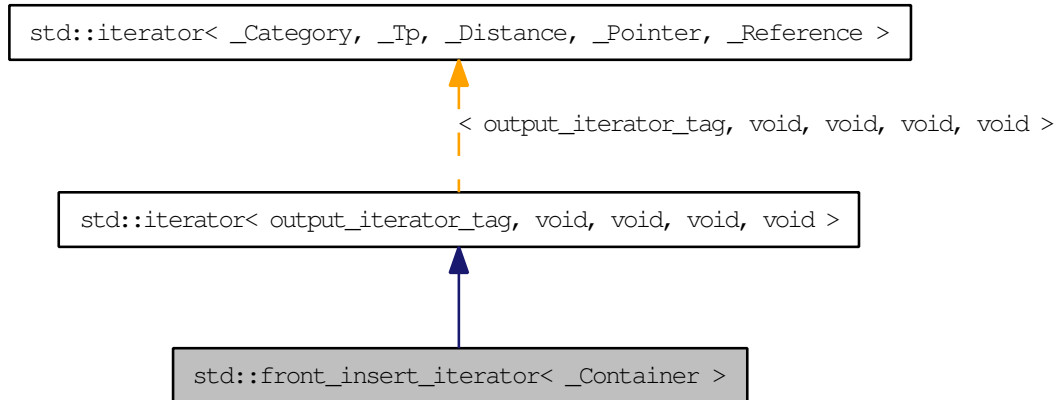
Definition at line 141 of file `postypes.h`.

The documentation for this class was generated from the following file:

- [postypes.h](#)

## 5.464 `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++` (int)
- `front_insert_iterator` & `operator++` ()
- `front_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_-value)
- `front_insert_iterator` & `operator=` (typename `_Container::const_reference` \_\_-value)

## Protected Attributes

- `_Container * container`

### 5.464.1 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 478 of file `stl_iterator.h`.

### 5.464.2 Member Typedef Documentation

**5.464.2.1** `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 486 of file `stl_iterator.h`.

**5.464.2.2** `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 115 of file `stl_iterator_base_types.h`.

**5.464.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 111 of file `stl_iterator_base_types.h`.

**5.464.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 117 of file `stl_iterator_base_types.h`.

## **5.464 std::front\_insert\_iterator< \_Container > Class Template Reference 2639**

**5.464.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file stl\_iterator\_base\_types.h.

**5.464.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file stl\_iterator\_base\_types.h.

### **5.464.3 Constructor & Destructor Documentation**

**5.464.3.1** `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 489 of file stl\_iterator.h.

### **5.464.4 Member Function Documentation**

**5.464.4.1** `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator*() [inline]`

Simply returns \*this.

Definition at line 520 of file stl\_iterator.h.

**5.464.4.2** `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 530 of file stl\_iterator.h.

**5.464.4.3** `template<typename _Container > front_insert_iterator& std::front_insert_iterator<_Container >::operator++ () [inline]`

Simply returns \*this. (This iterator does not *move*.)

Definition at line 525 of file stl\_iterator.h.

**5.464.4.4** `template<typename _Container > front_insert_iterator& std::front_insert_iterator<_Container >::operator= (typename _Container::const_reference __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 503 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

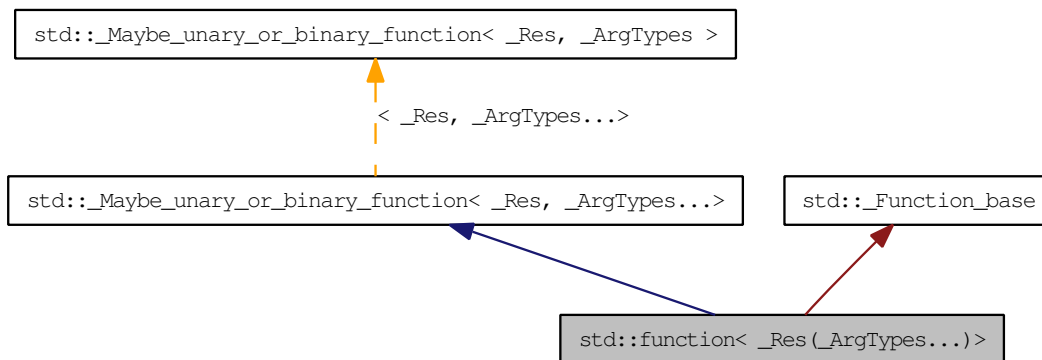
- [stl\\_iterator.h](#)



## 5.465 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

- `template<typename _Function >`  
`function` (`_Function` `__f`, `typename enable_if< !is_integral< _Function >::value, _Useless >::type` `=_Useless()`)
- `function` (`function` `&&__x`)
- `function` (`const function` `&__x`)
- `function` (`_M_clear_type` `*`)
- `function` (`()`)
- `operator bool` (`()` `const`)
- `template<typename _Res2, typename... _ArgTypes2>`  
`void operator!=` (`const function< _Res2(_ArgTypes2...)>` `&`) `const`
- `_Res` `operator()` (`_ArgTypes...__args`) `const`
- `template<typename _Function >`  
`enable_if<!is_integral< _Function >::value, function & >::type` `operator=`  
`(reference_wrapper< _Function > __f)`

- `template<typename _Functor >`  
`enable_if<!is_integral< _Functor >::value, function & >::type operator= (_-`  
`Functor &&__f)`
- `function & operator= (_M_clear_type *)`
- `function & operator= (function &&__x)`
- `function & operator= (const function &&__x)`
- `template<typename _Res2, typename... _ArgTypes2>`  
`void operator== (const function< _Res2(_ArgTypes2...)> &) const`
- `void swap (function &&__x)`
- `template<typename _Functor >`  
`const _Functor * target () const`
- `template<typename _Functor >`  
`_Functor * target ()`
- `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`

### 5.465.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::function< _Res(_-`  
`ArgTypes...)>`

Primary class template for `std::function`.

Polymorphic function wrapper.

Definition at line 1800 of file functional.

## 5.465.2 Constructor & Destructor Documentation

**5.465.2.1** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function () [inline, explicit]`

Default construct creates an empty function call wrapper.

**Postcondition:**

!(bool)\*this

Definition at line 1818 of file functional.

**5.465.2.2** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function (_M_clear_type *) [inline]`

Default construct creates an empty function call wrapper.

**Postcondition:**

!(bool)\*this

Definition at line 1824 of file functional.

**5.465.2.3** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function (const function< _Res(_ArgTypes...)> & _x) [inline]`

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 2068 of file functional.

**5.465.2.4** `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 1843 of file functional.

**5.465.2.5** `template<typename _Res , typename... _ArgTypes> template<typename _Function > std::function<_Res(_ArgTypes...)>::function (_Function _f, typename enable_if<!is_integral<_Function >::value, _Useless >::type = _Useless ()) [inline]`

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 2082 of file functional.

### 5.465.3 Member Function Documentation

**5.465.3.1** `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 2011 of file `functional`.

```
5.465.3.2 template<typename _Res , typename... _ArgTypes> _Res
std::function< _Res(_ArgTypes...)>::operator() (_ArgTypes...
__args) const [inline]
```

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 2100 of file `functional`.

```
5.465.3.3 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 1955 of file `functional`.

```
5.465.3.4 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 1946 of file `functional`.

**5.465.3.5** `template<typename _Res, typename... _ArgTypes> function& std::function<_Res(_ArgTypes...)>::operator= (_M_clear_type *) [inline]`

Function assignment to zero.

**Postcondition:**

`!(bool)*this`

**Returns:**

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 1917 of file `functional`.

**5.465.3.6** `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 1903 of file `functional`.

**5.465.3.7** `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 1885 of file functional.

**5.465.3.8** `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 1970 of file functional.

References `std::swap()`.

**5.465.3.9** `template<typename _Res , typename... _ArgTypes> template<typename _Functor > const _Functor * std::function< _Res(_ArgTypes...)>::target () const [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 2146 of file functional.

**5.465.3.10** `template<typename _Res , typename... _ArgTypes>  
template<typename _Functor > _Functor * std::function<  
_Res(_ArgTypes...)>::target () [inline]`

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 2127 of file functional.

**5.465.3.11** `template<typename _Res , typename... _ArgTypes> const type_info  
& std::function< _Res(_ArgTypes...)>::target_type () const  
[inline]`

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 2111 of file functional.

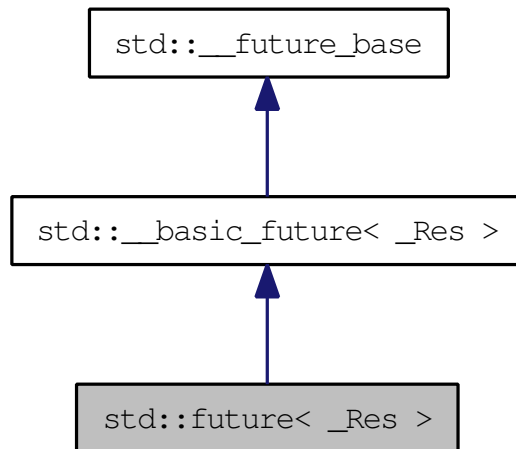
The documentation for this class was generated from the following file:

- [functional](#)



## 5.466 `std::future< _Res >` Class Template Reference

Primary template for `future`. Inheritance diagram for `std::future< _Res >`:



### Public Member Functions

- `future` (const `future` &)
- `future` (`future` &&\_uf)
- `_Res` `get` ()
- `future` & `operator=` (`future` &&\_fut)
- `future` & `operator=` (const `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

### 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 >`

### 5.466.1 Detailed Description

`template<typename _Res> class std::future< _Res >`

Primary template for `future`.

Definition at line 559 of file `future`.

### 5.466.2 Constructor & Destructor Documentation

**5.466.2.1** `template<typename _Res > std::future< _Res >::future (future< _Res > && __uf) [inline]`

Move constructor.

Definition at line 577 of file `future`.

### 5.466.3 Member Function Documentation

**5.466.3.1** `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 513 of file `future`.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

**5.466.3.2** `template<typename _Res > _Res std::future< _Res >::get () [inline]`

Retrieving the value.

Definition at line 591 of file future.

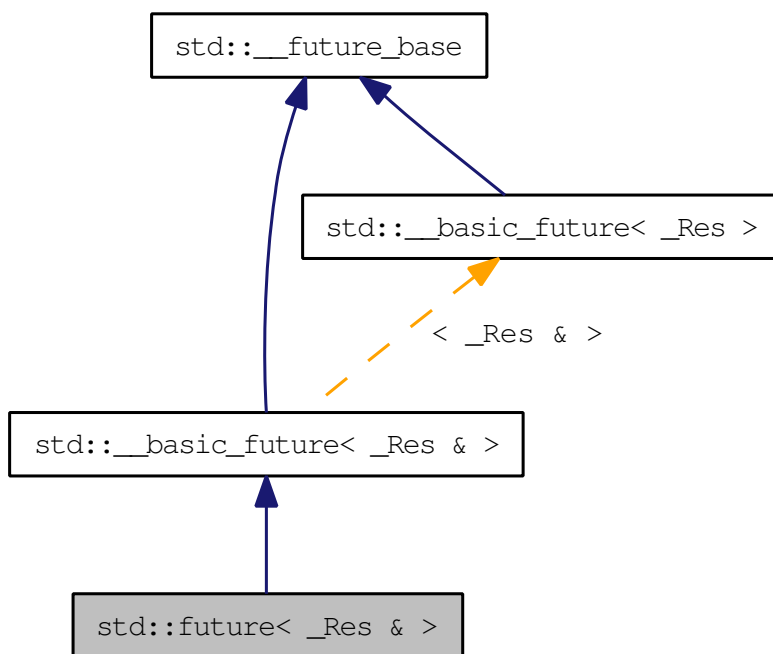
References `std::__basic_future<_Res>::_M_get_result()`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.467 `std::future<_Res &>` Class Template Reference

Partial specialization for `future<R&>`. Inheritance diagram for `std::future<_Res &>`:



### Public Member Functions

- **future** (const [future](#) &)
- **future** ([future](#) &&\_uf)
- `_Res` & **get** ()
- **future** & **operator=** ([future](#) &&\_fut)
- **future** & **operator=** (const [future](#) &)
- 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

### 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 &>`

### 5.467.1 Detailed Description

`template<typename _Res> class std::future<_Res &>`

Partial specialization for `future<R&>`.

Definition at line 600 of file `future`.

### 5.467.2 Constructor & Destructor Documentation

**5.467.2.1** `template<typename _Res > std::future<_Res &>::future (future<_Res &> && __uf) [inline]`

Move constructor.

Definition at line 618 of file `future`.

### 5.467.3 Member Function Documentation

**5.467.3.1** `__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 513 of file `future`.

References `std::rethrow_exception()`.

**5.467.3.2** `template<typename _Res > _Res& std::future< _Res & >::get ()`  
`[inline]`

Retrieving the value.

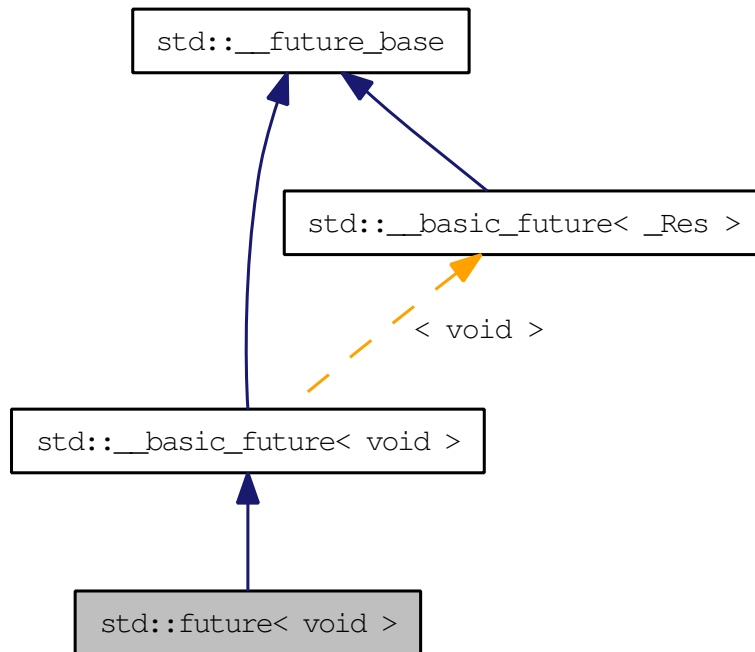
Definition at line 632 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.468 `std::future< void >` Class Template Reference

Explicit specialization for `future<void>`. Inheritance diagram for `std::future< void >`:



### Public Member Functions

- **future** (const `future` &)
- **future** (`future` &&\_uf)
- void **get** ()
- **future** & **operator=** (`future` &&\_fut)
- **future** & **operator=** (const `future` &)
- 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

### 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 >`

### 5.468.1 Detailed Description

`template<> class std::future< void >`

Explicit specialization for `future<void>`.

Definition at line 641 of file future.

### 5.468.2 Constructor & Destructor Documentation

**5.468.2.1** `std::future< void >::future (future< void > && __uf) [inline]`

Move constructor.

Definition at line 659 of file future.

### 5.468.3 Member Function Documentation

**5.468.3.1** `__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 513 of file future.

**5.468.3.2** `void std::future< void >::get () [inline]`

Retrieving the value.

Definition at line 673 of file future.



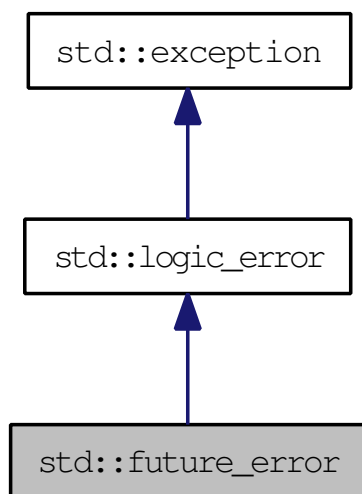
References `std::_basic_future<_Res >::_M_get_result()`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.469 `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` ()

### 5.469.1 Detailed Description

Exception type thrown by futures.

Definition at line 85 of file `future`.

### 5.469.2 Member Function Documentation

#### 5.469.2.1 `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).

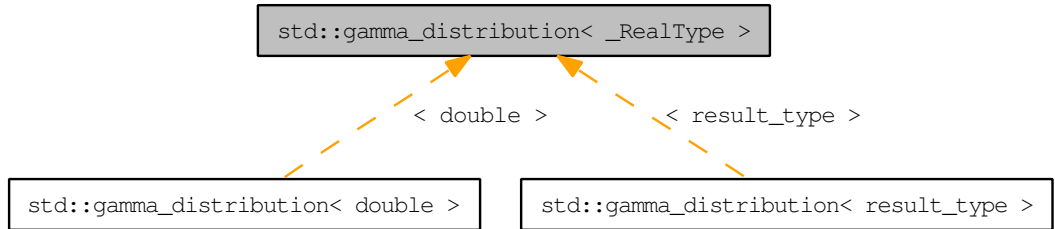
Reimplemented from `std::logic_error`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.470 `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** (const [param\\_type](#) &\_\_p)
- **gamma\_distribution** (`_RealType` \_\_alpha\_val=`_RealType`(1), `_RealType` \_\_beta\_val=`_RealType`(1))
- `_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, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & operator>>> (std::basic_istream<_-`  
`CharT, _Traits> &, std::gamma_distribution<_RealType1> &)`

### 5.470.1 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 2118 of file `random.h`.

### 5.470.2 Member Typedef Documentation

**5.470.2.1** `template<typename _RealType = double> typedef _RealType`  
`std::gamma_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2125 of file `random.h`.

### 5.470.3 Constructor & Destructor Documentation

**5.470.3.1** `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 2165 of file `random.h`.

## 5.470.4 Member Function Documentation

**5.470.4.1** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType >::alpha () const [inline]`

Returns the  $\alpha$  of the distribution.

Definition at line 2186 of file random.h.

**5.470.4.2** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType >::beta () const [inline]`

Returns the  $\beta$  of the distribution.

Definition at line 2193 of file random.h.

**5.470.4.3** `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 2222 of file random.h.

Referenced by `std::gamma_distribution< result_type >::max()`.

**5.470.4.4** `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 2215 of file random.h.

**5.470.4.5** `template<typename _RealType > template<typename  
_UniformRandomNumberGenerator > gamma_distribution<  
_RealType >::result_type std::gamma_distribution<_RealType  
>::operator() (_UniformRandomNumberGenerator & __urng, const  
param_type & __param) [inline]`

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 1945 of file random.tcc.

References `std::log()`, and `std::pow()`.

**5.470.4.6** `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 2208 of file `random.h`.

**5.470.4.7** `template<typename _RealType = double> param_type  
std::gamma_distribution<_RealType>::param () const [inline]`

Returns the parameter `set` of the distribution.

Definition at line 2200 of file `random.h`.

**5.470.4.8** `template<typename _RealType = double> void  
std::gamma_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Definition at line 2179 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()`.

## 5.470.5 Friends And Related Function Documentation

**5.470.5.1** `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.

**5.470.5.2** `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](#)



## 5.471 std::gamma\_distribution<\_RealType>::param\_type Struct Reference

### 5.471 std::gamma\_distribution<\_RealType>::param\_type Struct Reference

#### Public Types

- typedef [gamma\\_distribution<\\_RealType>](#) **distribution\_type**

#### 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>](#)

#### 5.471.1 Detailed Description

**template<typename \_RealType = double> struct std::gamma\_distribution<\_RealType>::param\_type**

Parameter type.

Definition at line 2127 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.472 `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` (const [param\\_type](#) &\_\_p)
- `geometric_distribution` (double \_\_p=0.5)
- `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` ()

#### 5.472.1 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 3194 of file `random.h`.

## 5.472.2 Member Typedef Documentation

**5.472.2.1** `template<typename _IntType = int> typedef _IntType  
std::geometric_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3201 of file `random.h`.

## 5.472.3 Member Function Documentation

**5.472.3.1** `template<typename _IntType = int> result_type  
std::geometric_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3283 of file `random.h`.

**5.472.3.2** `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 3276 of file `random.h`.

**5.472.3.3** `template<typename _IntType = int> double  
std::geometric_distribution< _IntType >::p () const [inline]`

Returns the distribution parameter `p`.

Definition at line 3254 of file `random.h`.

**5.472.3.4** `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.

Definition at line 3269 of file `random.h`.

**5.472.3.5** `template<typename _IntType = int> param_type  
std::geometric_distribution< _IntType >::param () const  
[inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 3261 of file random.h.

Referenced by `std::operator>>()`.

**5.472.3.6** `template<typename _IntType = int> void  
std::geometric_distribution< _IntType >::reset () [inline]`

Resets the distribution state. Does nothing for the geometric distribution.

Definition at line 3248 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.473 std::geometric\_distribution< \_IntType >::param\_type Struct Reference

### 5.473 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 >](#)

#### 5.473.1 Detailed Description

**template<typename \_IntType = int> struct std::geometric\_distribution< \_IntType >::param\_type**

Parameter type.

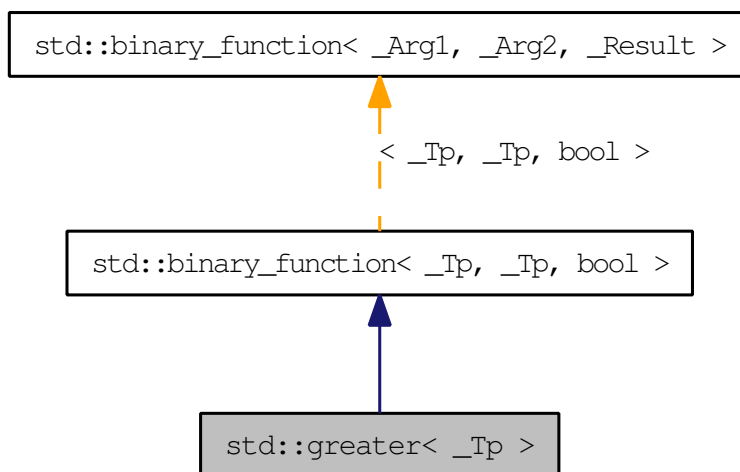
Definition at line 3203 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.474 `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`

#### 5.474.1 Detailed Description

```
template<typename _Tp> struct std::greater< _Tp >
```

One of the [comparison functors](#).

Definition at line 217 of file `stl_function.h`.

## 5.474.2 Member Typedef Documentation

**5.474.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, bool >::first_argument_type` `[inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.474.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool >::result_type` `[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.474.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool >::second_argument_type` `[inherited]`

the type of the second argument

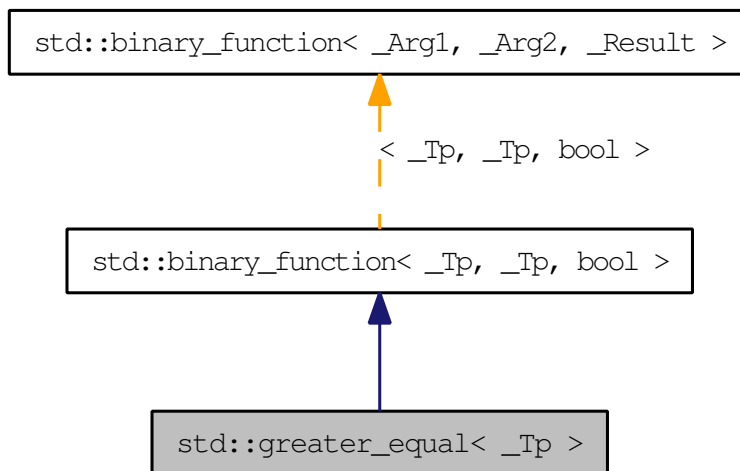
Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.475 `std::greater_equal< _Tp >` Struct Template Reference

One of the [comparison functors](#). 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`

#### 5.475.1 Detailed Description

`template<typename _Tp> struct std::greater_equal< _Tp >`

One of the [comparison functors](#).

Definition at line 235 of file `stl_function.h`.



## 5.475.2 Member Typedef Documentation

**5.475.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.475.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.475.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.476 `std::gslice` Class Reference

Class defining multi-dimensional subset of an [array](#).

### Public Member Functions

- [gslice](#) (const [gslice](#) &)
- [gslice](#) (size\_t, const [valarray](#)< size\_t > &, const [valarray](#)< size\_t > &)
- [gslice](#) ()
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- [size\\_t](#) [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

### Friends

- class [valarray](#)

#### 5.476.1 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 63 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 5.477 `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
- `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
- `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
- `template<class _Dom >`  
void `operator=` (const `_Expr< _Dom, _Tp >` &) const
- void `operator=` (const `_Tp` &) const
- void `operator=` (const `valarray< _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`

## Friends

- `class valarray< _Tp >`

### 5.477.1 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.

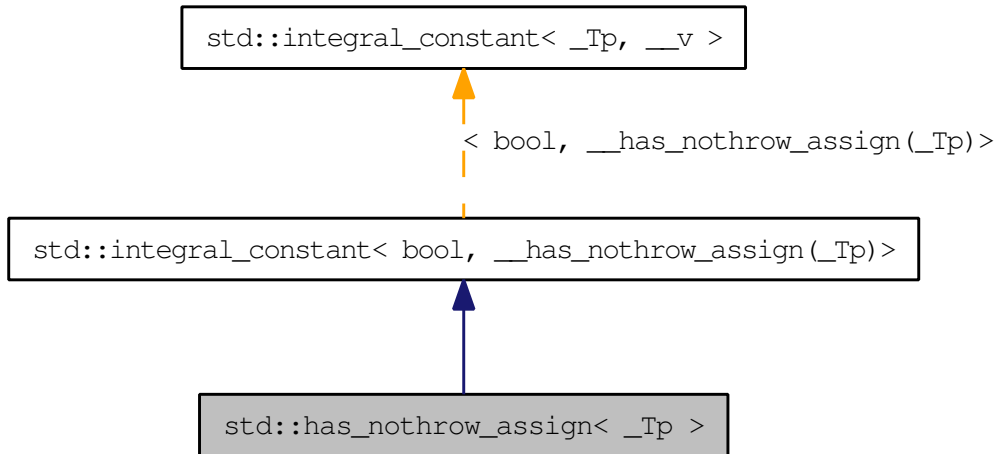
Definition at line 59 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice\\_array.h](#)

## 5.478 `std::has_nothrow_assign< _Tp >` Struct Template Reference

[has\\_nothrow\\_assign](#) Inheritance diagram for `std::has_nothrow_assign< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.478.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_assign< _Tp >
```

[has\\_nothrow\\_assign](#)

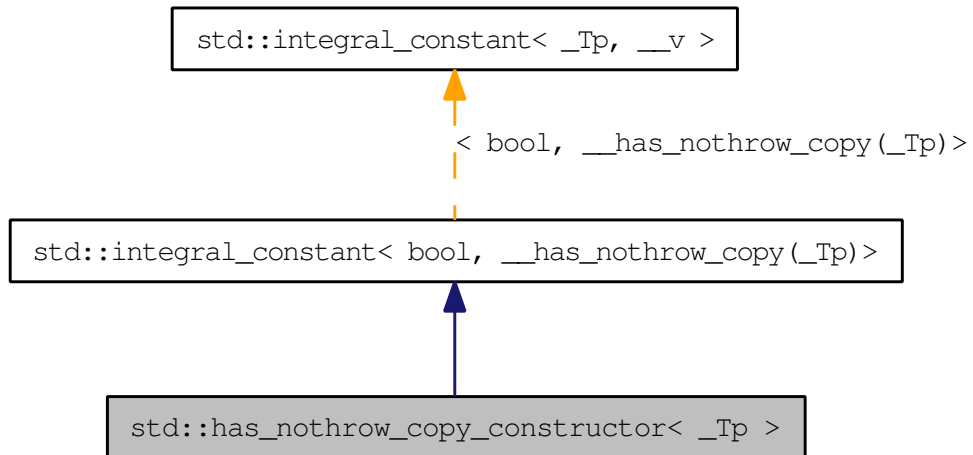
Definition at line 275 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.479 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.479.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_constructor< _Tp >
```

[has\\_nothrow\\_copy\\_constructor](#)

Definition at line 269 of file `type_traits`.

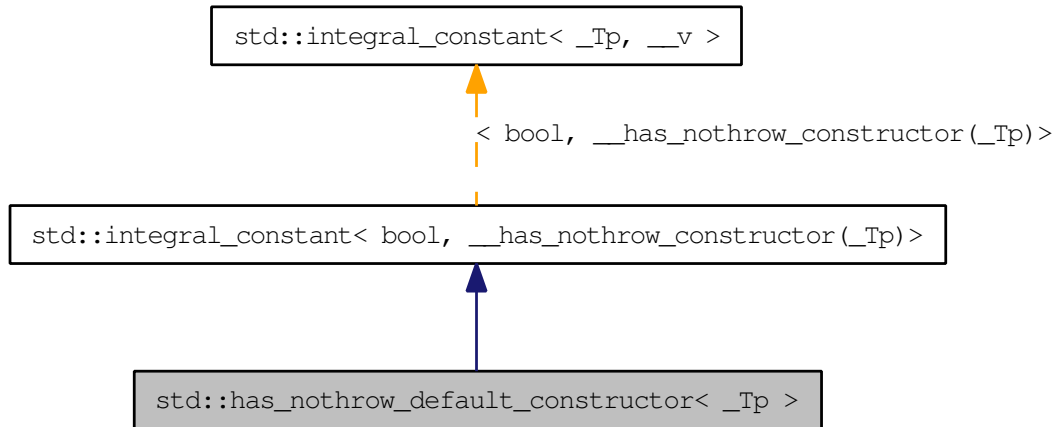
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.480 `std::has_nothrow_default_constructor< _Tp >` Struct Template Reference 2679

### 5.480 `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**

#### Static Public Attributes

- static const `bool` **value**

#### 5.480.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_default_constructor< _Tp >
```

[has\\_nothrow\\_default\\_constructor](#)

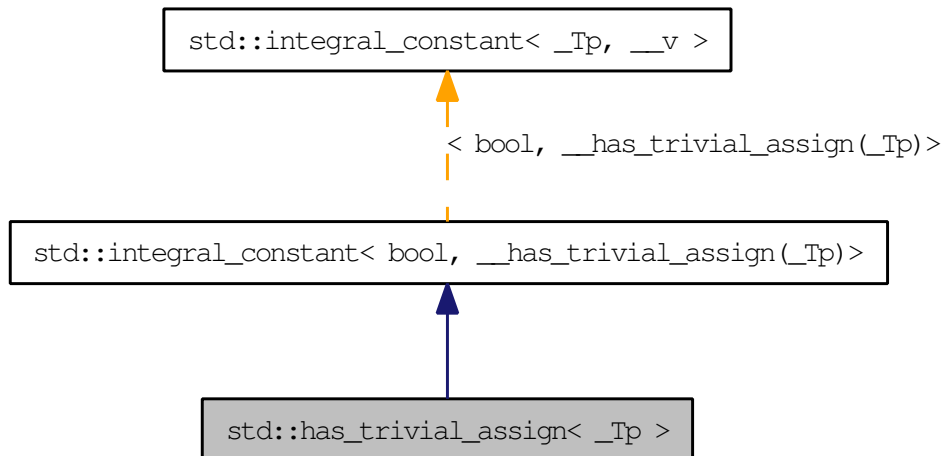
Definition at line 263 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.481 `std::has_trivial_assign< _Tp >` Struct Template Reference

[has\\_trivial\\_assign](#) Inheritance diagram for `std::has_trivial_assign< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)`< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.481.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_assign< _Tp >`

[has\\_trivial\\_assign](#)

Definition at line 251 of file `type_traits`.

The documentation for this struct was generated from the following file:

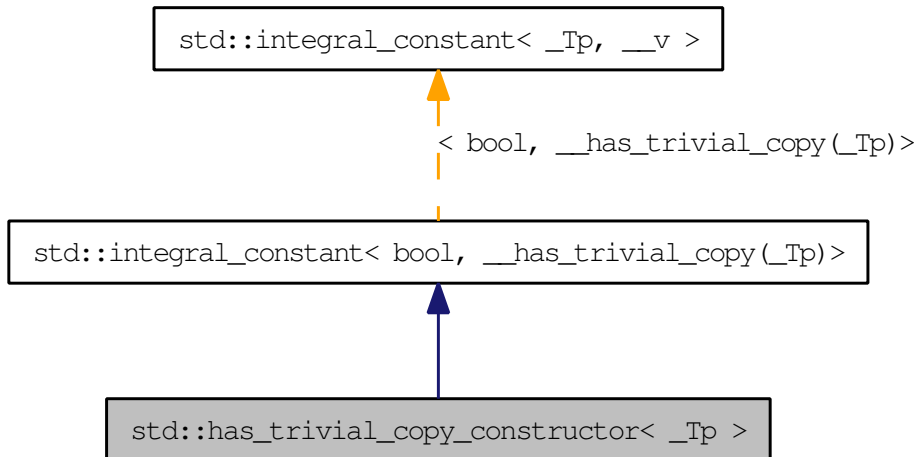
- [type\\_traits](#)



## 5.482 `std::has_trivial_copy_constructor< _Tp >` Struct Template Reference 2681

### 5.482 `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**

#### Static Public Attributes

- static const `bool` **value**

#### 5.482.1 Detailed Description

```
template<typename _Tp> struct std::has_trivial_copy_constructor< _Tp >
```

[has\\_trivial\\_copy\\_constructor](#)

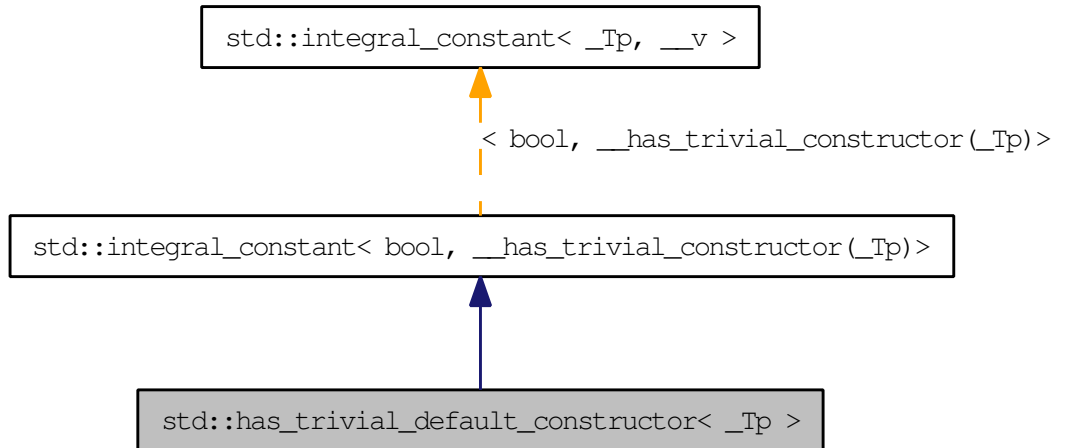
Definition at line 245 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.483 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.483.1 Detailed Description

```
template<typename _Tp> struct std::has_trivial_default_constructor< _Tp >
```

[has\\_trivial\\_default\\_constructor](#)

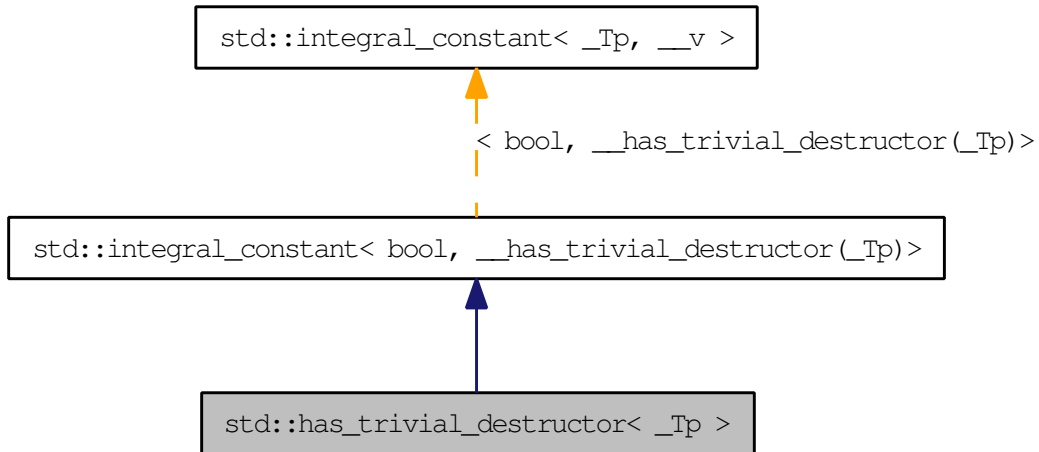
Definition at line 239 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.484 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.484.1 Detailed Description

```
template<typename _Tp> struct std::has_trivial_destructor< _Tp >
```

[has\\_trivial\\_destructor](#)

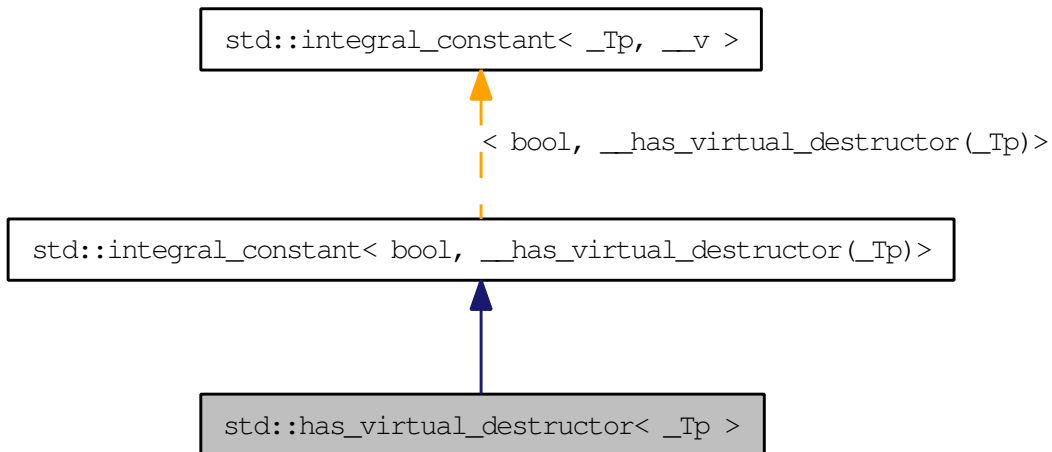
Definition at line 257 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.485 `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**

### Static Public Attributes

- static const bool **value**

### 5.485.1 Detailed Description

`template<typename _Tp> struct std::has_virtual_destructor< _Tp >`

[has\\_virtual\\_destructor](#)

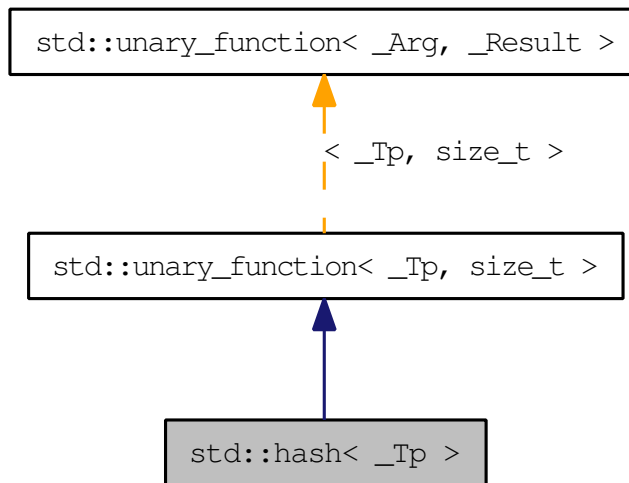
Definition at line 344 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.486 std::hash< \_Tp > Struct Template Reference

Primary class template [hash](#). Inheritance diagram for std::hash< \_Tp >:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `size_t` [result\\_type](#)

### Public Member Functions

- `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`
- `size_t operator() (_Tp __val) const`

### 5.486.1 Detailed Description

`template<typename _Tp> struct std::hash< _Tp >`

Primary class template [hash](#).

Definition at line 50 of file `functional_hash.h`.

### 5.486.2 Member Typedef Documentation

**5.486.2.1** `typedef _Tp std::unary_function< _Tp , size_t >::argument_type`  
[`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.486.2.2** `typedef size_t std::unary_function< _Tp , size_t >::result_type`  
[`inherited`]

`result_type` is the return type

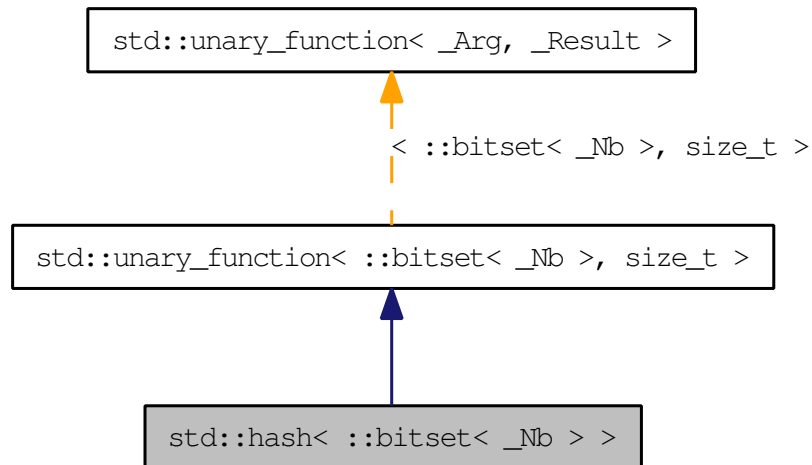
Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.487 `std::hash< ::bitset< _Nb > >` Struct Template Reference

`std::hash` specialization for `bitset`. Inheritance diagram for `std::hash< ::bitset< _Nb > >`:



### Public Types

- typedef `::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const ::bitset< _Nb > &__b) const`

#### 5.487.1 Detailed Description

```
template<size_t _Nb> struct std::hash< ::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 1498 of file `bitset`.



## 5.487.2 Member Typedef Documentation

**5.487.2.1** `typedef ::bitset< _Nb > std::unary_function< ::bitset< _Nb > , size_t >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.487.2.2** `typedef size_t std::unary_function< ::bitset< _Nb > , size_t >::result_type [inherited]`

`result_type` is the return type

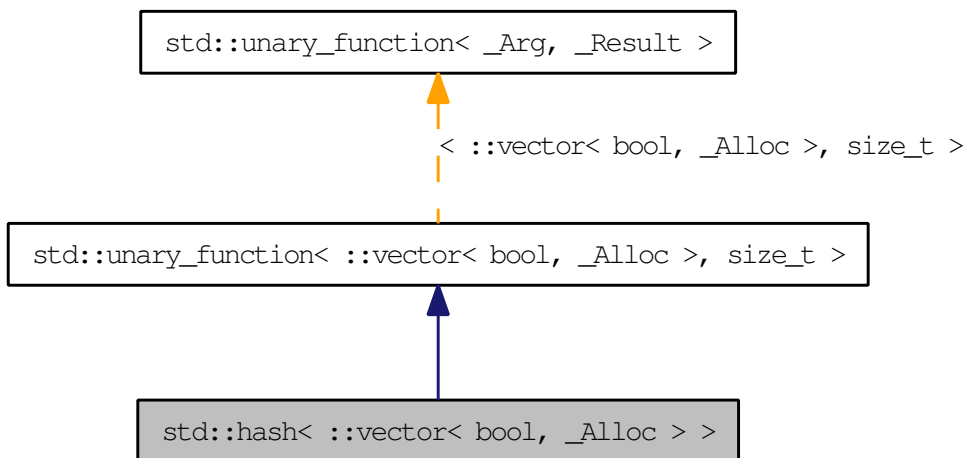
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.488 `std::hash< ::vector< bool, _Alloc > >` Struct Template Reference

`std::hash` specialization for `vector<bool>`. Inheritance diagram for `std::hash< ::vector< bool, _Alloc > >`:



### Public Types

- typedef `::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (`const ::vector< bool, _Alloc > &_b`) `const`

#### 5.488.1 Detailed Description

```
template<typename _Alloc> struct std::hash< ::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 1040 of file `stl_bvector.h`.

## 5.488.2 Member Typedef Documentation

**5.488.2.1** `typedef ::vector< bool, _Alloc > std::unary_function< ::vector< bool, _Alloc >, size_t>::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.488.2.2** `typedef size_t std::unary_function< ::vector< bool, _Alloc >, size_t>::result_type [inherited]`

`result_type` is the return type

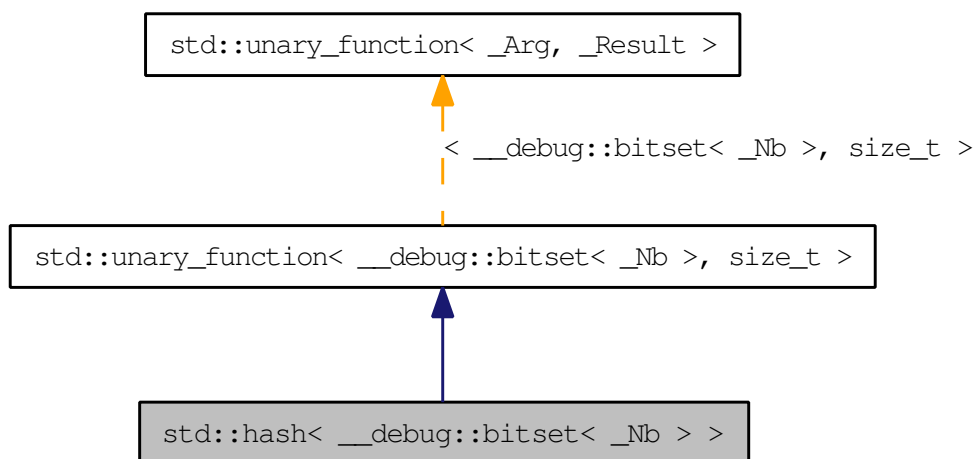
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_bvector.h](#)

## 5.489 `std::hash< __debug::bitset< _Nb > >` Struct Template Reference

`std::hash` specialization for `bitset`. Inheritance diagram for `std::hash< __debug::bitset< _Nb > >`:



### Public Types

- typedef `__debug::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const __debug::bitset< _Nb > &__b) const`

#### 5.489.1 Detailed Description

```
template<size_t _Nb> struct std::hash< __debug::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 387 of file `debug/bitset`.

## 5.489.2 Member Typedef Documentation

**5.489.2.1** `typedef __debug::bitset< _Nb > std::unary_function< __debug::bitset< _Nb >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.489.2.2** `typedef size_t std::unary_function< __debug::bitset< _Nb >, size_t >::result_type [inherited]`

`result_type` is the return type

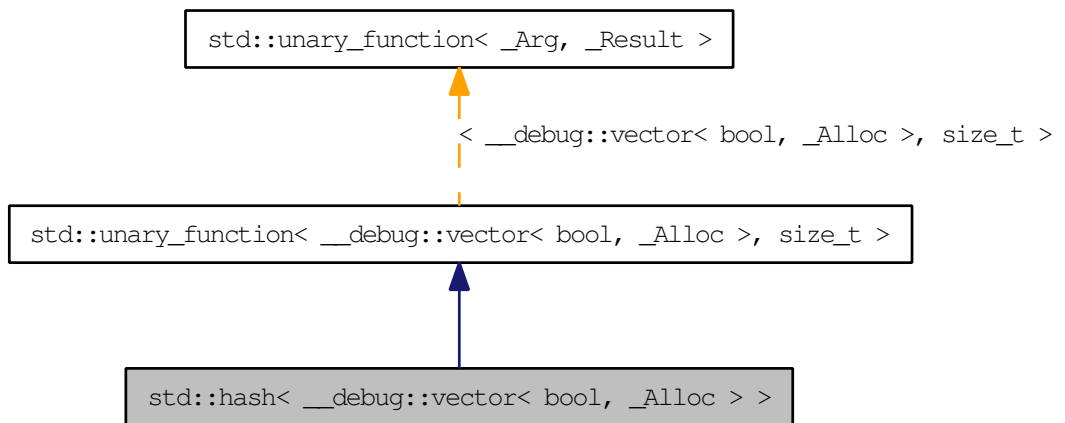
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

## 5.490 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

`std::hash` specialization for `vector<bool>`. Inheritance diagram for `std::hash< __debug::vector< bool, _Alloc > >`:



### Public Types

- typedef `__debug::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (const `__debug::vector< bool, _Alloc > &__b`) const

#### 5.490.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __debug::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 544 of file `debug/vector`.

## 5.490 std::hash< \_\_debug::vector< bool, \_Alloc > > Struct Template Reference

### 5.490.2 Member Typedef Documentation

**5.490.2.1** `typedef __debug::vector< bool, _Alloc > std::unary_function< __debug::vector< bool, _Alloc >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.490.2.2** `typedef size_t std::unary_function< __debug::vector< bool, _Alloc >, size_t >::result_type [inherited]`

`result_type` is the return type

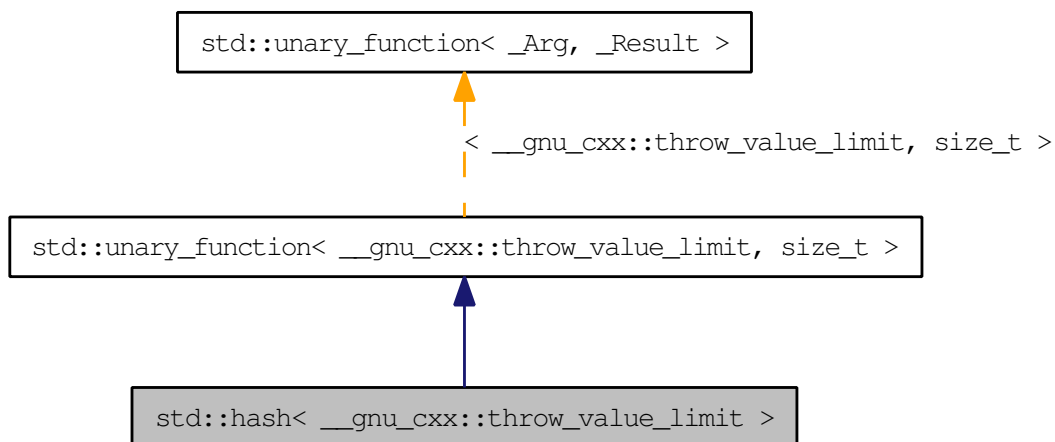
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)

## 5.491 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`. 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`

#### 5.491.1 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 728 of file `throw_allocator.h`.



## 5.491 std::hash< \_\_gnu\_cxx::throw\_value\_limit > Struct Template Reference 2697

### 5.491.2 Member Typedef Documentation

**5.491.2.1** `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 (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.491.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]`

`result_type` is the return type

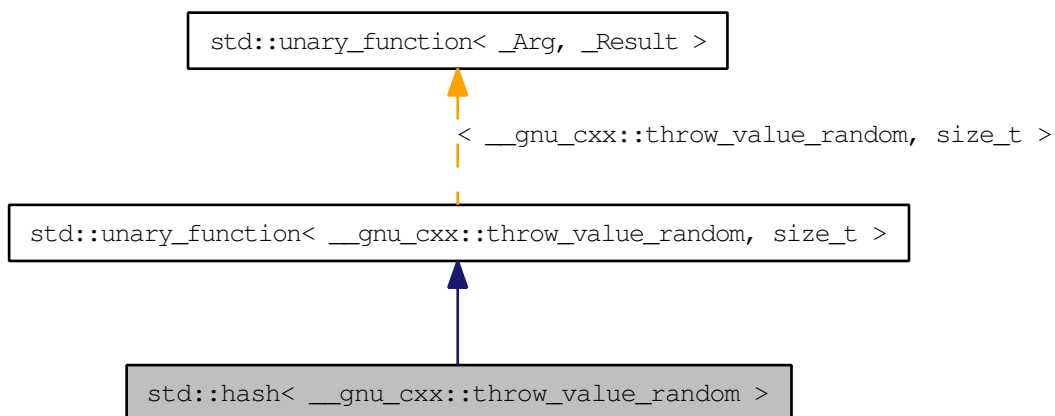
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.492 `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`

#### 5.492.1 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 742 of file `throw_allocator.h`.

## 5.492 std::hash< \_\_gnu\_cxx::throw\_value\_random > Struct Template Reference

### 5.492.2 Member Typedef Documentation

**5.492.2.1** `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 (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.492.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type`  
[`inherited`]

`result_type` is the return type

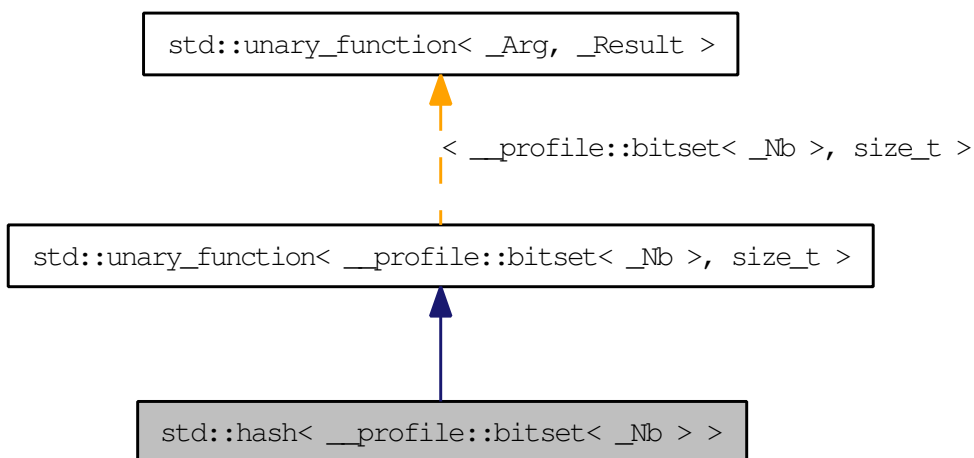
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.493 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

`std::hash` specialization for `bitset`. Inheritance diagram for `std::hash< __profile::bitset< _Nb > >`:



### Public Types

- typedef `__profile::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const __profile::bitset< _Nb > &__b) const`

#### 5.493.1 Detailed Description

```
template<size_t _Nb> struct std::hash< __profile::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 361 of file `profile/bitset`.

## **5.493.2 Member Typedef Documentation**

**5.493.2.1 typedef \_\_profile::bitset< \_Nb > std::unary\_function< \_\_profile::bitset< \_Nb >, size\_t >::argument\_type [inherited]**

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.493.2.2 typedef size\_t std::unary\_function< \_\_profile::bitset< \_Nb >, size\_t >::result\_type [inherited]**

`result_type` is the return type

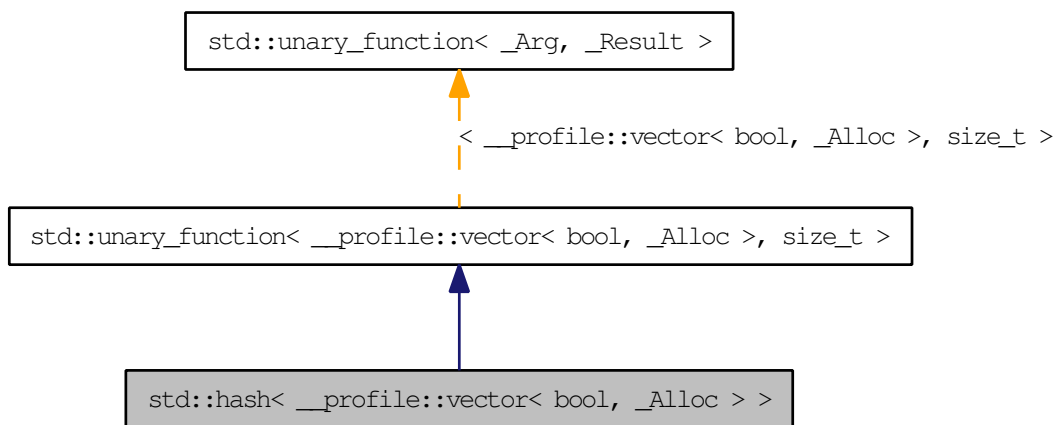
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [profile/bitset](#)

## 5.494 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

`std::hash` specialization for `vector<bool>`. Inheritance diagram for `std::hash< __profile::vector< bool, _Alloc > >`:



### Public Types

- typedef `__profile::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (`const __profile::vector< bool, _Alloc > &__b`) `const`

#### 5.494.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __profile::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 469 of file `profile/vector`.

## **5.494 std::hash< \_\_profile::vector< bool, \_Alloc > > Struct Template Reference**

### **5.494.2 Member Typedef Documentation**

**5.494.2.1** `typedef __profile::vector< bool, _Alloc > std::unary_function< __profile::vector< bool, _Alloc >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.494.2.2** `typedef size_t std::unary_function< __profile::vector< bool, _Alloc >, size_t >::result_type [inherited]`

`result_type` is the return type

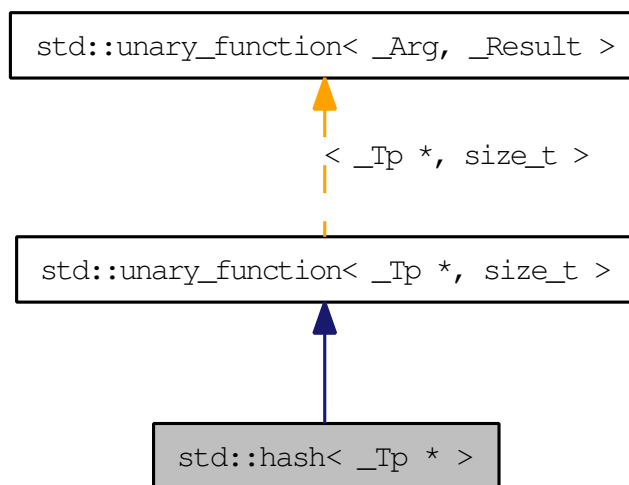
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [profile/vector](#)

## 5.495 `std::hash<_Tp*>` Struct Template Reference

Partial specializations for pointer types. Inheritance diagram for `std::hash<_Tp*>`:



### Public Types

- typedef `_Tp*` [argument\\_type](#)
- typedef `size_t` [result\\_type](#)

### Public Member Functions

- `size_t operator() (_Tp* __p) const`

#### 5.495.1 Detailed Description

```
template<typename _Tp> struct std::hash<_Tp*>
```

Partial specializations for pointer types.

Definition at line 58 of file `functional_hash.h`.



## 5.495.2 Member Typedef Documentation

### 5.495.2.1 `typedef _Tp * std::unary_function<_Tp *, size_t >::argument_type` [`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.495.2.2 `typedef size_t std::unary_function<_Tp *, size_t >::result_type` [`inherited`]

`result_type` is the return type

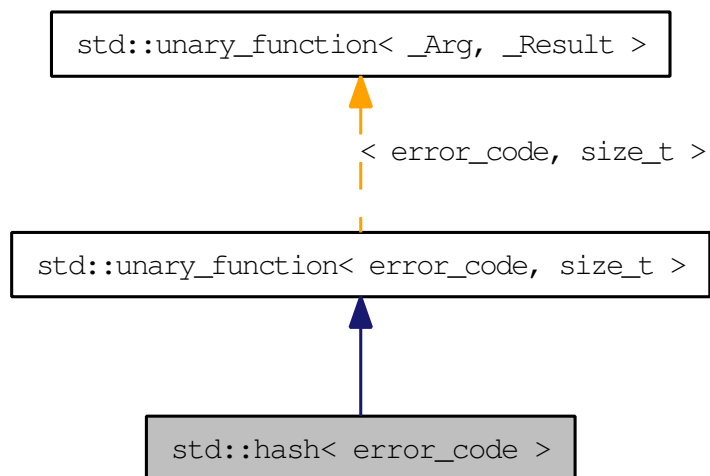
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.496 `std::hash< error_code >` Struct Template Reference

`std::hash` specialization for `error_code`. Inheritance diagram for `std::hash< error_code >`:



### Public Types

- typedef `error_code` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const error_code &__e) const`

#### 5.496.1 Detailed Description

`template<> struct std::hash< error_code >`

`std::hash` specialization for `error_code`.

Definition at line 352 of file `system_error`.

## 5.496.2 Member Typedef Documentation

### 5.496.2.1 `typedef error_code std::unary_function< error_code , size_t >::argument_type [inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.496.2.2 `typedef size_t std::unary_function< error_code , size_t >::result_type [inherited]`

`result_type` is the return type

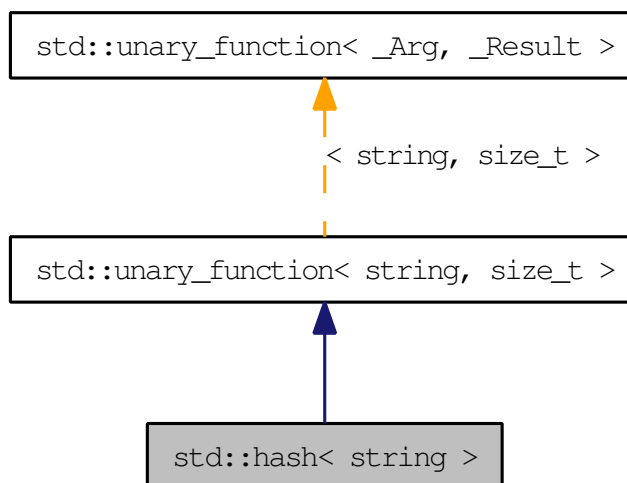
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.497 `std::hash< string >` Struct Template Reference

`std::hash` specialization for string. Inheritance diagram for `std::hash< string >`:



### Public Types

- typedef `string` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const string &__s) const`

#### 5.497.1 Detailed Description

`template<> struct std::hash< string >`

`std::hash` specialization for string.

Definition at line 2885 of file `basic_string.h`.

## 5.497.2 Member Typedef Documentation

### 5.497.2.1 typedef string std::unary\_function< string , size\_t >::argument\_type [inherited]

argument\_type is the type of the argument (no surprises here)

Definition at line 102 of file stl\_function.h.

### 5.497.2.2 typedef size\_t std::unary\_function< string , size\_t >::result\_type [inherited]

result\_type is the return type

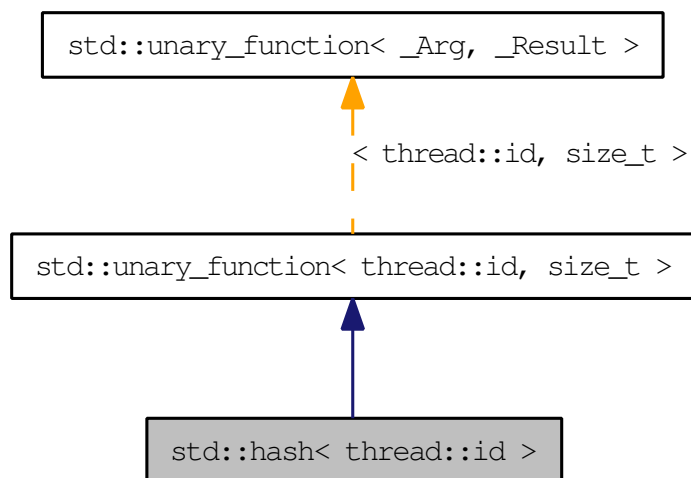
Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.498 `std::hash< thread::id >` Struct Template Reference

`std::hash` specialization for `thread::id`. Inheritance diagram for `std::hash< thread::id >`:



### Public Types

- typedef `thread::id` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (const `thread::id` &`__id`) const

#### 5.498.1 Detailed Description

`template<> struct std::hash< thread::id >`

`std::hash` specialization for `thread::id`.

Definition at line 226 of file `thread`.

## 5.498.2 Member Typedef Documentation

### 5.498.2.1 `typedef thread::id std::unary_function< thread::id , size_t >::argument_type` [inherited]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.498.2.2 `typedef size_t std::unary_function< thread::id , size_t >::result_type` [inherited]

`result_type` is the return type

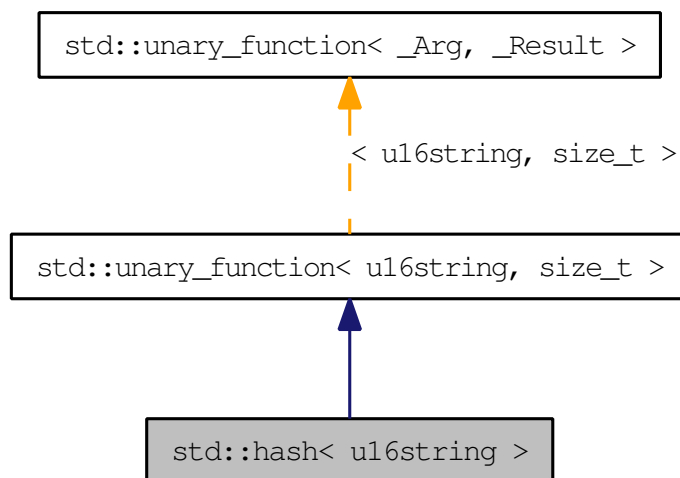
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [thread](#)

## 5.499 `std::hash< u16string >` Struct Template Reference

`std::hash` specialization for `u16string`. Inheritance diagram for `std::hash< u16string >`:



### Public Types

- typedef `u16string` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const u16string &__s) const`

#### 5.499.1 Detailed Description

```
template<> struct std::hash< u16string >
```

`std::hash` specialization for `u16string`.

Definition at line 2910 of file `basic_string.h`.



## 5.499.2 Member Typedef Documentation

### 5.499.2.1 typedef u16string std::unary\_function< u16string , size\_t >::argument\_type [inherited]

argument\_type is the type of the argument (no surprises here)

Definition at line 102 of file stl\_function.h.

### 5.499.2.2 typedef size\_t std::unary\_function< u16string , size\_t >::result\_type [inherited]

result\_type is the return type

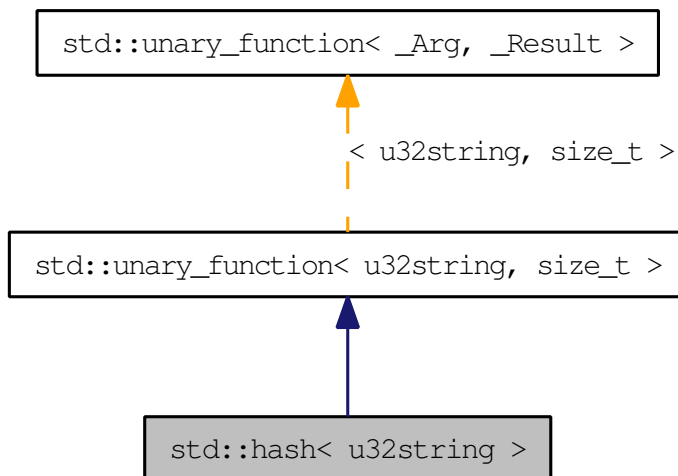
Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.500 `std::hash< u32string >` Struct Template Reference

`std::hash` specialization for `u32string`. Inheritance diagram for `std::hash< u32string >`:



### Public Types

- typedef `u32string` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const u32string &__s) const`

#### 5.500.1 Detailed Description

`template<> struct std::hash< u32string >`

`std::hash` specialization for `u32string`.

Definition at line 2921 of file `basic_string.h`.

## 5.500.2 Member Typedef Documentation

### 5.500.2.1 typedef u32string std::unary\_function< u32string , size\_t >::argument\_type [inherited]

argument\_type is the type of the argument (no surprises here)

Definition at line 102 of file stl\_function.h.

### 5.500.2.2 typedef size\_t std::unary\_function< u32string , size\_t >::result\_type [inherited]

result\_type is the return type

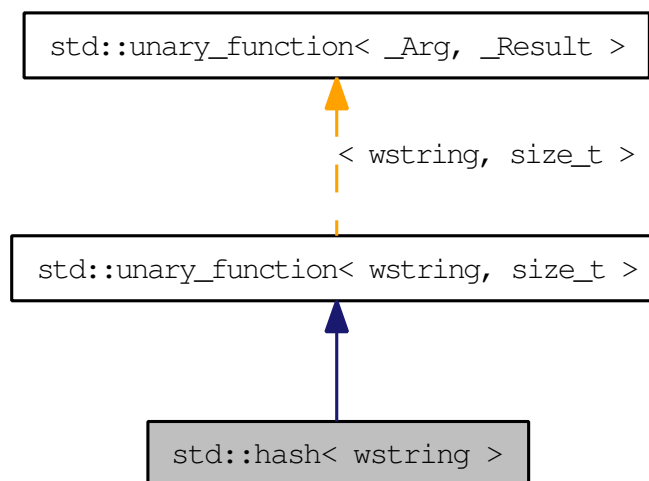
Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.501 `std::hash<wstring>` Struct Template Reference

`std::hash` specialization for `wstring`. Inheritance diagram for `std::hash<wstring>`:



### Public Types

- typedef `wstring` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const wstring &__s) const`

#### 5.501.1 Detailed Description

`template<> struct std::hash<wstring>`

`std::hash` specialization for `wstring`.

Definition at line 2896 of file `basic_string.h`.

## 5.501.2 Member Typedef Documentation

### 5.501.2.1 typedef wstring std::unary\_function< wstring , size\_t >::argument\_type [inherited]

argument\_type is the type of the argument (no surprises here)

Definition at line 102 of file stl\_function.h.

### 5.501.2.2 typedef size\_t std::unary\_function< wstring , size\_t >::result\_type [inherited]

result\_type is the return type

Definition at line 105 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.502 `std::identity<_Tp >` Struct Template Reference

[identity](#)

### Public Types

- `typedef _Tp type`

#### 5.502.1 Detailed Description

`template<typename _Tp> struct std::identity<_Tp >`

[identity](#)

Definition at line 44 of file `move.h`.

The documentation for this struct was generated from the following file:

- [move.h](#)

## 5.503 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType` Class Template Reference

### Public Types

- typedef `_UIntType` `result_type`

### Public Member Functions

- `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` (`result_type` `__s`)
- `independent_bits_engine` (`_RandomNumberEngine` &&`__rne`)
- `independent_bits_engine` (`const _RandomNumberEngine` &`__rne`)
- `independent_bits_engine` ()
- `const _RandomNumberEngine` & `base` () `const`
- void `discard` (`unsigned long long` `__z`)
- `result_type` `max` () `const`
- `result_type` `min` () `const`
- `result_type` `operator()` ()
- `template<typename _Sseq >`  
void `seed` (`_Sseq` &`_q`)
- void `seed` (`result_type` `__s`)
- void `seed` ()

### 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`)

### 5.503.1 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 921 of file random.h.

### 5.503.2 Member Typedef Documentation

**5.503.2.1** `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 930 of file random.h.

### 5.503.3 Constructor & Destructor Documentation

**5.503.3.1** `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 937 of file random.h.

**5.503.3.2** `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 947 of file random.h.



**5.503** `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>` **Class Template Reference** 2721

---

**5.503.3.3** `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 957 of file `random.h`.

**5.503.3.4** `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 967 of file `random.h`.

**5.503.3.5** `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 980 of file `random.h`.

### 5.503.4 Member Function Documentation

**5.503.4.1** `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 1015 of file random.h.

**5.503.4.2** `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.

#### Todo

Look for a faster way to do discard.

Definition at line 1042 of file random.h.

**5.503.4.3** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::max () const [inline]`

Gets the maximum value in the generated random number range.

#### Todo

This should be constexpr.

Definition at line 1033 of file random.h.

**5.503.4.4** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::min () const [inline]`

Gets the minimum value in the generated random number range.

#### Todo

This should be constexpr.

**5.503 std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType > Class Template Reference** 2723

---

Definition at line 1024 of file random.h.

**5.503.4.5** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType > independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::operator() () [inline]`

Gets the next value in the generated random number sequence.

Definition at line 709 of file random.tcc.

References std::log().

**5.503.4.6** `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 1007 of file random.h.

**5.503.4.7** `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 997 of file random.h.

**5.503.4.8** `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 989 of file random.h.

### 5.503.5 Friends And Related Function Documentation

**5.503.5.1** `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 two objects are equal, false otherwise.

Definition at line 1066 of file `random.h`.

**5.503.5.2** `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 1084 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.504 `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
- `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
- `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
- `template<class _Dom >`  
void `operator=` (const `_Expr< _Dom, _Tp >` &) const
- void `operator=` (const `_Tp` &) const
- void `operator=` (const `valarray< _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`

## Friends

- `class gslice_array< _Tp >`
- `class valarray< _Tp >`

### 5.504.1 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 61 of file `indirect_array.h`.

The documentation for this class was generated from the following file:

- [indirect\\_array.h](#)

## 5.505 `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

- `const_iterator` **begin** () const
- `const_iterator` **end** () const
- `size_type` **size** () const

#### 5.505.1 Detailed Description

```
template<class _E> class std::initializer_list<_E >
```

[initializer\\_list](#)

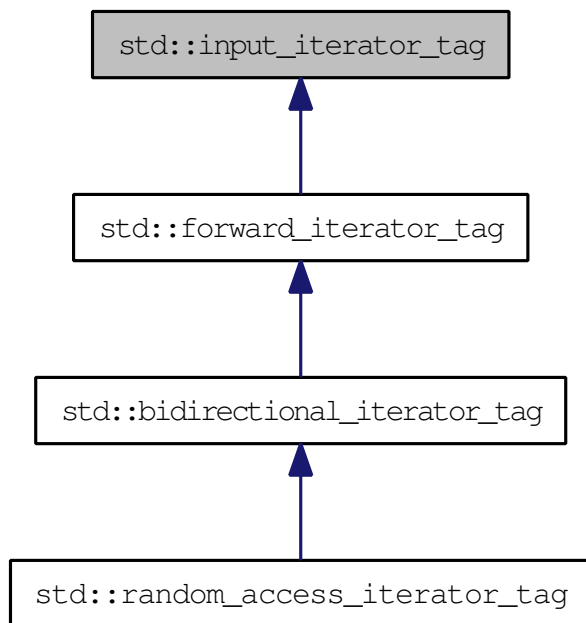
Definition at line 45 of file `initializer_list`.

The documentation for this class was generated from the following file:

- [initializer\\_list](#)

## 5.506 std::input\_iterator\_tag Struct Reference

Marking input iterators. Inheritance diagram for std::input\_iterator\_tag:



### 5.506.1 Detailed Description

Marking input iterators.

Definition at line 79 of file `stl_iterator_base_types.h`.

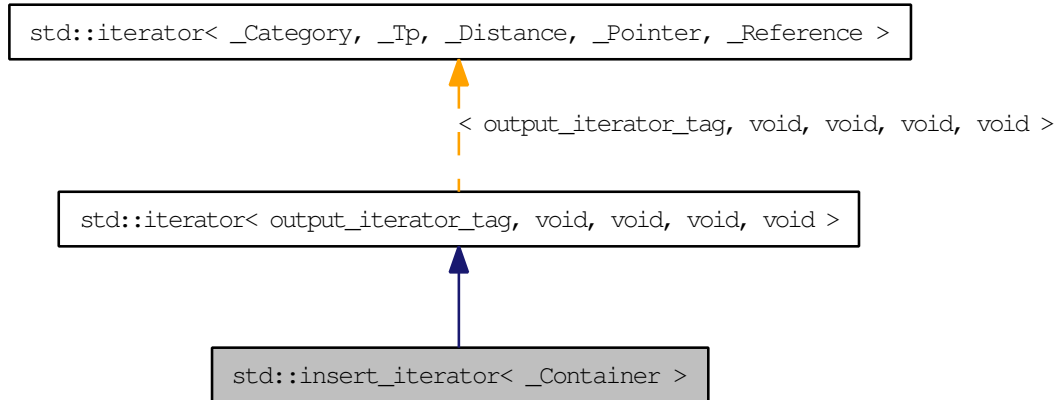
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)



## 5.507 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++` (int)
- `insert_iterator` & `operator++` ()
- `insert_iterator` & `operator=` (`typename _Container::value_type &&__value`)
- `insert_iterator` & `operator=` (`typename _Container::const_reference __value`)

### Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

### 5.507.1 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 565 of file stl\_iterator.h.

### 5.507.2 Member Typedef Documentation

**5.507.2.1 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 574 of file stl\_iterator.h.

**5.507.2.2 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 115 of file stl\_iterator\_base\_types.h.

**5.507.2.3 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void  
, void , void , void >::iterator\_category [inherited]**

One of the [tag types](#).

Definition at line 111 of file stl\_iterator\_base\_types.h.

**5.507.2.4 typedef void std::iterator< output\_iterator\_tag , void , void , void ,  
void >::pointer [inherited]**

This type represents a pointer-to-value\_type.

Definition at line 117 of file stl\_iterator\_base\_types.h.

**5.507.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file `stl_iterator_base_types.h`.

**5.507.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file `stl_iterator_base_types.h`.

### 5.507.3 Constructor & Destructor Documentation

**5.507.3.1** `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 580 of file `stl_iterator.h`.

### 5.507.4 Member Function Documentation

**5.507.4.1** `template<typename _Container> insert_iterator& std::insert_iterator<_Container >::operator* () [inline]`

Simply returns `*this`.

Definition at line 626 of file `stl_iterator.h`.

**5.507.4.2** `template<typename _Container> insert_iterator& std::insert_iterator<_Container >::operator++ (int) [inline]`

Simply returns `*this`. (This iterator does not *move*.).

Definition at line 636 of file `stl_iterator.h`.

**5.507.4.3** `template<typename _Container> insert_iterator&  
std::insert_iterator< _Container >::operator++ () [inline]`

Simply returns \*this. (This iterator does not *move*).

Definition at line 631 of file stl\_iterator.h.

**5.507.4.4** `template<typename _Container> insert_iterator&  
std::insert_iterator< _Container >::operator= (typename  
_Container::const_reference __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 607 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.508 `std::integral_constant< _Tp, __v >` Struct Template Reference

### [integral\\_constant](#)

Inherited by `std::__is_integral_helper< typename >`, `std::__is_member_object_pointer_helper< typename >`, `std::__is_signed_helper< _Tp, bool, bool >`, `std::__is_void_helper< typename >`, `std::is_bind_expression< _Bind< _Signature > >`, `std::is_bind_expression< _Bind_result< _Result, _Signature > >`, `std::is_const< typename >`, `std::is_const< _Tp const >`, `std::is_error_code_enum< _Tp >`, `std::is_error_code_enum< future_errc >`, `std::is_error_condition_enum< errc >`, `std::is_function< typename >`, `std::is_function< _Res(_ArgTypes...) const volatile >`, `std::is_function< _Res(_ArgTypes...) >`, `std::is_function< _Res(_ArgTypes.....) const >`, `std::is_function< _Res(_ArgTypes.....) volatile >`, `std::is_rvalue_reference< typename >`, and `std::is_rvalue_reference< _Tp && >`.

### Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

### Static Public Attributes

- static const `_Tp` **value**

### 5.508.1 Detailed Description

```
template<typename _Tp, _Tp __v> struct std::integral_constant< _Tp, __v >
```

#### [integral\\_constant](#)

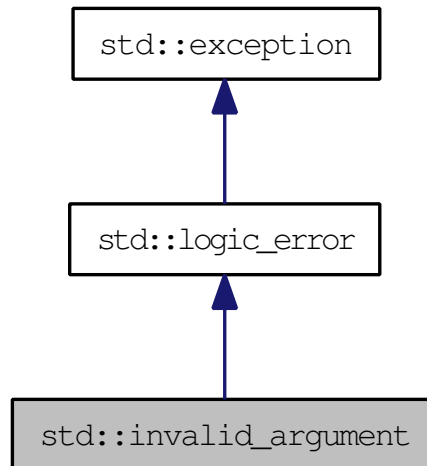
Definition at line 67 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.509 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 ()

### 5.509.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 80 of file `stdexcept`.

### 5.509.2 Member Function Documentation

#### 5.509.2.1 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](#).

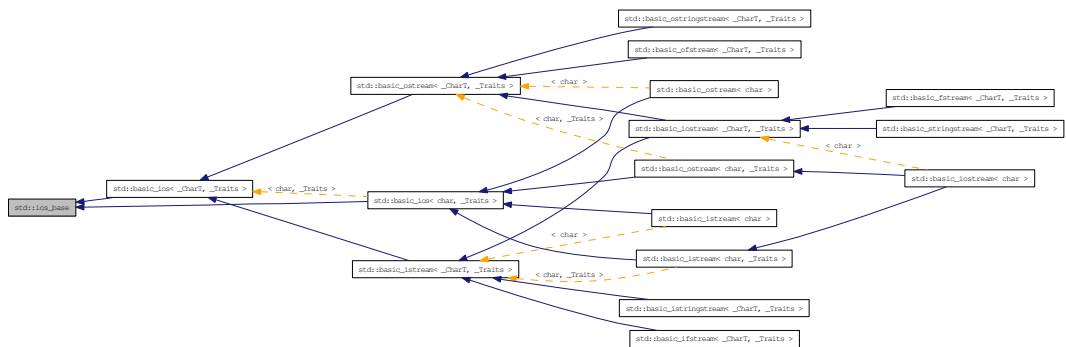
The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.510 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 (fmtflags __fmtfl)`
- `fmtflags flags () const`
- `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 __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `void unsetf (fmtflags __mask)`
- `streamsize width (streamsize __wide)`
- `streamsize width () 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\\_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](#)
- [iostate \\_M\\_exception](#)
- [fmtflags \\_M\\_flags](#)
- [locale \\_M\\_ios\\_locale](#)
- [\\_Words \\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- [streamsize \\_M\\_precision](#)
- [iostate \\_M\\_streambuf\\_state](#)
- [streamsize \\_M\\_width](#)
- [\\_Words \\* \\_M\\_word](#)
- [int \\_M\\_word\\_size](#)
- [\\_Words \\_M\\_word\\_zero](#)

### 5.510.1 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 207 of file `ios_base.h`.

### 5.510.2 Member Typedef Documentation

#### 5.510.2.1 `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 444 of file `ios_base.h`.

#### 5.510.2.2 `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 263 of file ios\_base.h.

#### 5.510.2.3 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 338 of file ios\_base.h.

#### 5.510.2.4 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 369 of file ios\_base.h.

#### 5.510.2.5 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 401 of file ios\_base.h.

### 5.510.3 Member Enumeration Documentation

#### 5.510.3.1 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 427 of file ios\_base.h.

### 5.510.4 Constructor & Destructor Documentation

#### 5.510.4.1 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).

### 5.510.5 Member Function Documentation

#### 5.510.5.1 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 705 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()`.

**5.510.5.2 `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 561 of file `ios_base.h`.

**5.510.5.3 `fmtflags std::ios_base::flags () const [inline]`**

Access to format flags.

**Returns:**

The format control flags for both input and output.

Definition at line 550 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<<()`, and `std::operator>>()`.

**5.510.5.4 `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 694 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()`.

**5.510.5.5 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>`.

**5.510.5.6 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 740 of file `ios_base.h`.

**5.510.5.7** `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 629 of file `ios_base.h`.

**5.510.5.8** `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 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.510.5.9** `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 761 of file `ios_base.h`.



**5.510.5.10 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.

**5.510.5.11 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 594 of file `ios_base.h`.

**5.510.5.12 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 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

#### 5.510.5.13 `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>

#### 5.510.5.14 `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 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nunitbuf()`, and `std::nouppercase()`.

#### 5.510.5.15 `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 652 of file `ios_base.h`.

**5.510.5.16 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 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.510.5.17 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 `word` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `word` 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 `word` and `pword` arrays.

**5.510.6 Member Data Documentation****5.510.6.1 const fmtflags std::ios\_base::adjustfield [static]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

#### 5.510.6.2 `const openmode std::ios_base::app` **[static]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

#### 5.510.6.3 `const openmode std::ios_base::ate` **[static]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

#### 5.510.6.4 `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 342 of file `ios_base.h`.

Referenced by `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::operator<<()`, `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 >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, `std::basic_istream<_CharT, _Traits >::unget()`, and `std::basic_ostream<_CharT, _Traits >::write()`.

#### 5.510.6.5 `const fmtflags std::ios_base::basefield` **[static]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.510.6.6 const seekdir std::ios\_base::beg [static]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**5.510.6.7 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 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**5.510.6.8 const fmtflags std::ios\_base::boolalpha [static]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.510.6.9 const seekdir std::ios\_base::cur [static]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), 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().

**5.510.6.10 const fmtflags std::ios\_base::dec [static]**

Converts integer input or generates integer output in `decimal` base.

Definition at line 269 of file ios\_base.h.

**5.510.6.11 const seekdir std::ios\_base::end [static]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 5.510.6.12 `const iostate std::ios_base::eofbit` [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 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_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 >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, and `std::ws()`.

#### 5.510.6.13 `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 350 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_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `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()`, and `std::basic_ostream<_CharT, _Traits >::seekp()`.

#### 5.510.6.14 `const fmtflags std::ios_base::fixed` [static]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

#### 5.510.6.15 `const fmtflags std::ios_base::floatfield` [static]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios\_base.h.

#### 5.510.6.16 const iostate std::ios\_base::goodbit [static]

Indicates all is well.

Definition at line 353 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 >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 5.510.6.17 const fmtflags std::ios\_base::hex [static]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.510.6.18 const openmode std::ios\_base::in [static]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios\_base.h.

Referenced by 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\_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().

**5.510.6.19 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 280 of file `ios_base.h`.

**5.510.6.20 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 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

**5.510.6.21 const fmtflags std::ios\_base::oct [static]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.510.6.22 const openmode std::ios\_base::out [static]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf<_CharT, _Traits >::overflow()`, `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()`.

**5.510.6.23 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 291 of file `ios_base.h`.

**5.510.6.24 const fmtflags std::ios\_base::scientific [static]**

Generates floating-point output in scientific notation.



Definition at line 294 of file `ios_base.h`.

#### 5.510.6.25 `const fmtflags std::ios_base::showbase [static]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 298 of file `ios_base.h`.

#### 5.510.6.26 `const fmtflags std::ios_base::showpoint [static]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file `ios_base.h`.

#### 5.510.6.27 `const fmtflags std::ios_base::showpos [static]`

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

#### 5.510.6.28 `const fmtflags std::ios_base::skipws [static]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

#### 5.510.6.29 `const openmode std::ios_base::trunc [static]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

#### 5.510.6.30 `const fmtflags std::ios_base::unitbuf [static]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

#### 5.510.6.31 `const fmtflags std::ios_base::uppercase [static]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

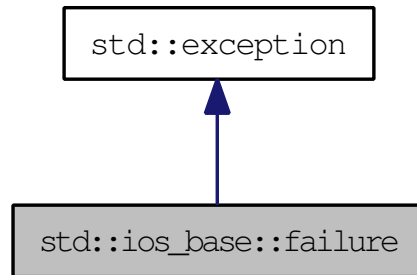
The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 5.511 `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 ()

#### 5.511.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

Definition at line 217 of file `ios_base.h`.

#### 5.511.2 Member Function Documentation

##### 5.511.2.1 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.

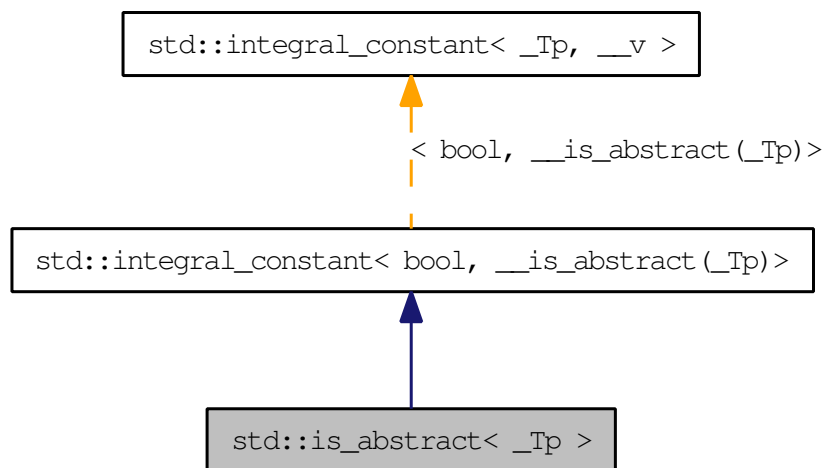
Reimplemented from `std::exception`.

The documentation for this class was generated from the following file:

- `ios_base.h`

## 5.512 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.512.1 Detailed Description

```
template<typename _Tp> struct std::is_abstract< _Tp >
```

[is\\_abstract](#)

Definition at line 338 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.513 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**

### Static Public Attributes

- static const bool **value**

#### 5.513.1 Detailed Description

**template<typename \_Tp> struct std::is\_arithmetic< \_Tp >**

[is\\_arithmetic](#)

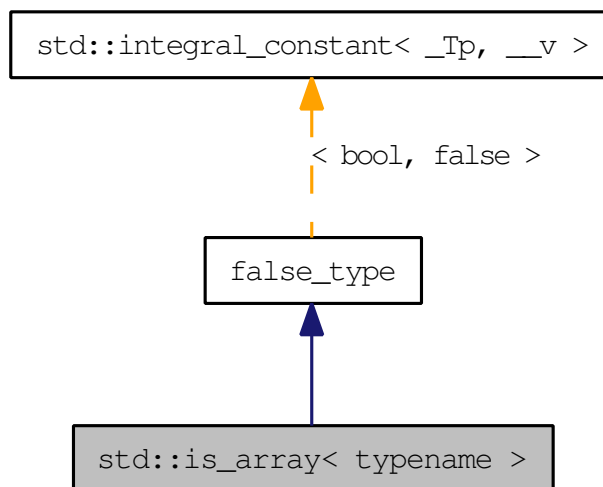
Definition at line 255 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.514 `std::is_array< typename >` Struct Template Reference

[is\\_array](#) Inheritance diagram for `std::is_array< typename >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Static Public Attributes

- static const bool **value**

#### 5.514.1 Detailed Description

`template<typename> struct std::is_array< typename >`

[is\\_array](#)

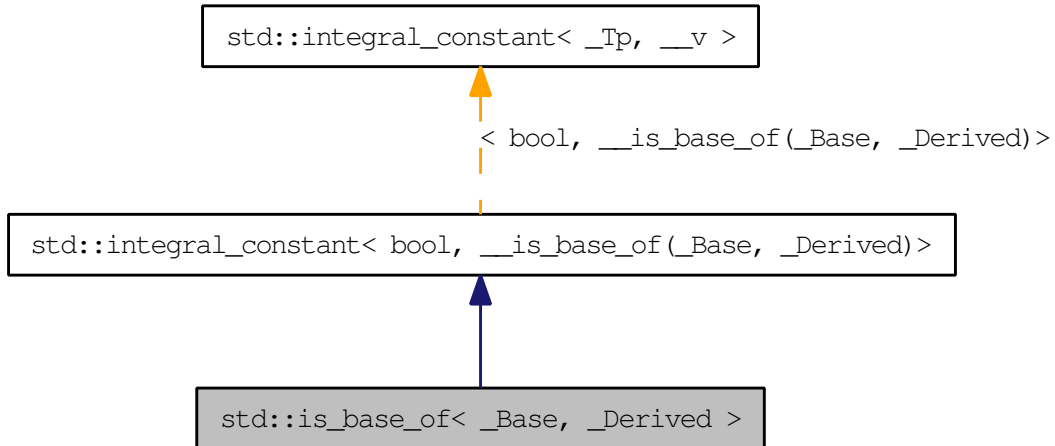
Definition at line 147 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.515 `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**

### Static Public Attributes

- static const `bool` **value**

### 5.515.1 Detailed Description

```
template<typename _Base, typename _Derived> struct std::is_base_of<_Base,
_Derived >
```

[is\\_base\\_of](#)

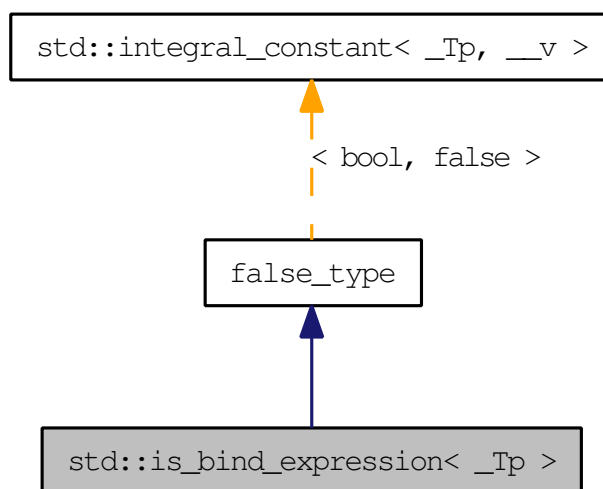
Definition at line 283 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.516 `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<bool, __v>` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.516.1 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 778 of file `functional`.

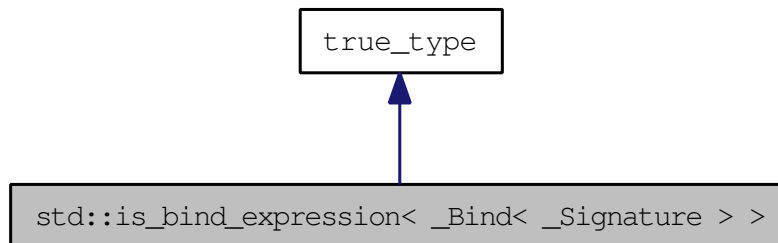
The documentation for this struct was generated from the following file:



- [functional](#)

## 5.517 `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`

### Static Public Attributes

- static const `_Tp value`

#### 5.517.1 Detailed Description

`template<typename _Signature> struct std::is_bind_expression< _Bind< _Signature > >`

Class template `_Bind` is always a bind expression.

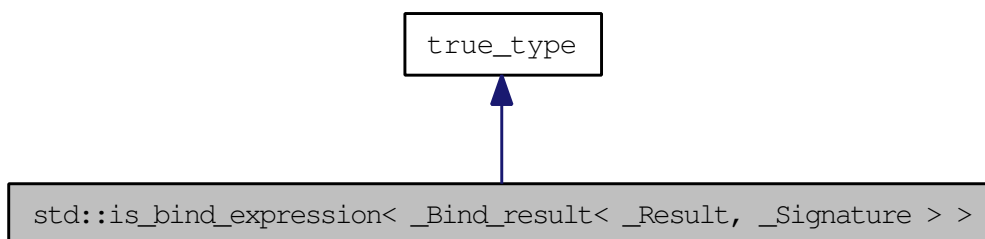
Definition at line 1354 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

## 5.518 `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**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.518.1 Detailed Description

```
template<typename _Result, typename _Signature> struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind` is always a bind expression.

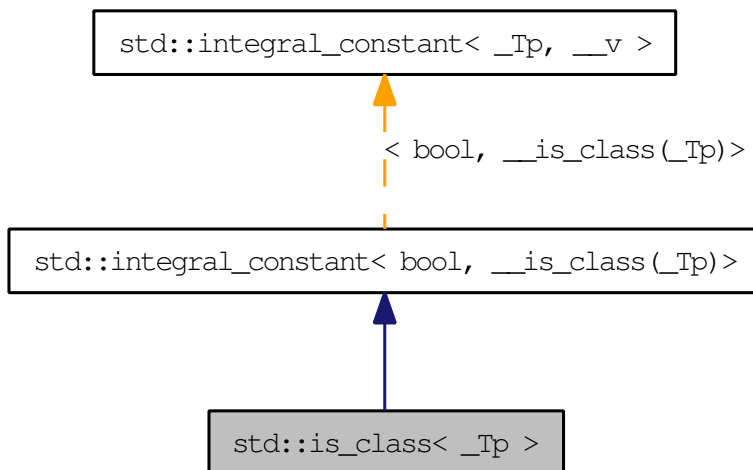
Definition at line 1362 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.519 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.519.1 Detailed Description

```
template<typename _Tp> struct std::is_class< _Tp >
```

[is\\_class](#)

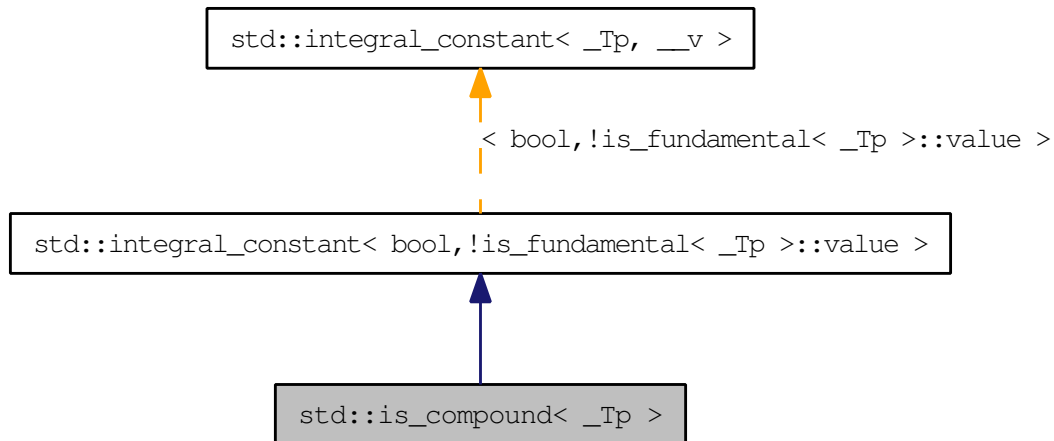
Definition at line 218 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.520 `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**

### Static Public Attributes

- static const `bool` **value**

### 5.520.1 Detailed Description

```
template<typename _Tp> struct std::is_compound<_Tp>
```

`is_compound`

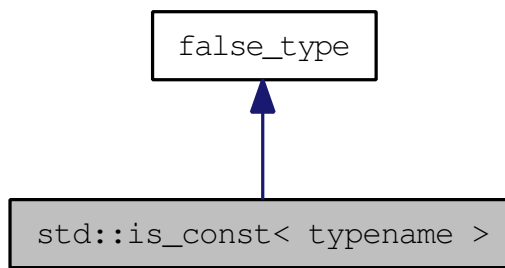
Definition at line 290 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- `tr1_impl/type_traits`

## 5.521 `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**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.521.1 Detailed Description

`template<typename> struct std::is_const< typename >`

[is\\_const](#)

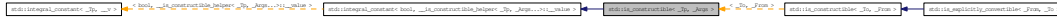
Definition at line 308 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.522 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.522.1 Detailed Description

`template<typename _Tp, typename... _Args> struct std::is_constructible< _Tp, _Args >`

[is\\_constructible](#)

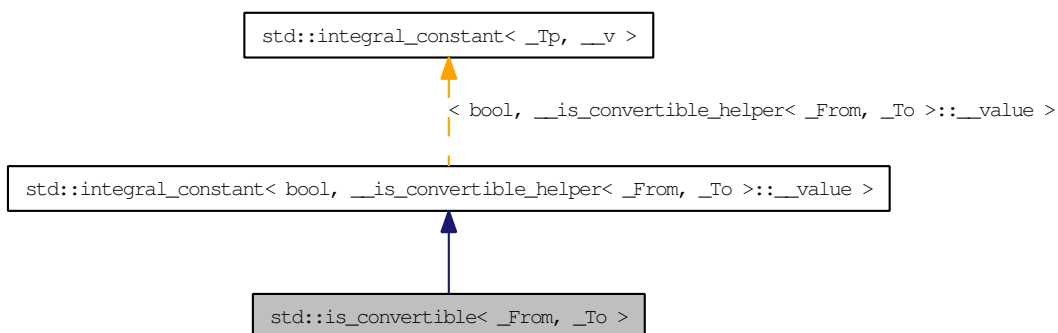
Definition at line 231 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.523 `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**

### Static Public Attributes

- static const bool **value**

#### 5.523.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_convertible< _From,
_To >
```

[is\\_convertible](#)

Definition at line 309 of file `type_traits`.

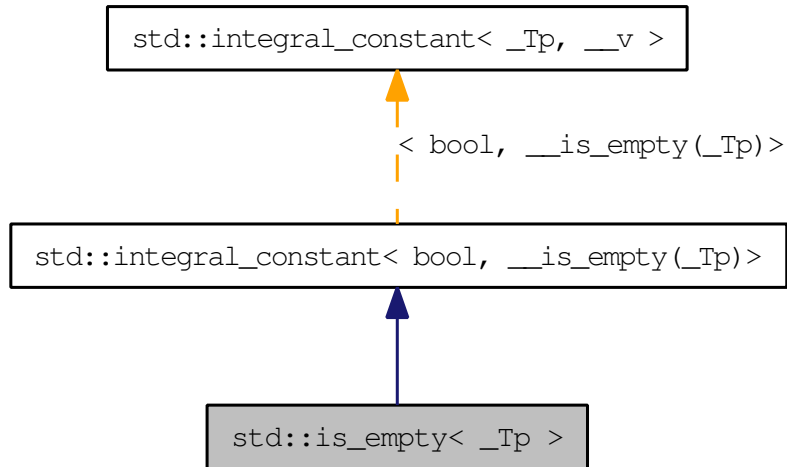
The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 5.524 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.524.1 Detailed Description

```
template<typename _Tp> struct std::is_empty< _Tp >
```

[is\\_empty](#)

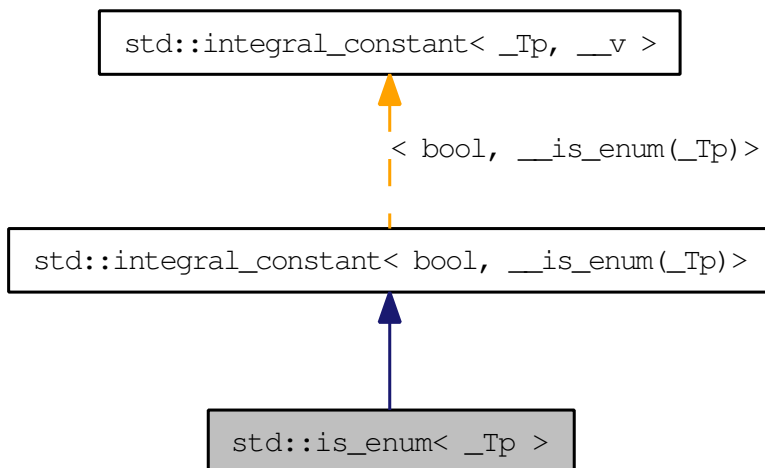
Definition at line 326 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.525 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.525.1 Detailed Description

```
template<typename _Tp> struct std::is_enum< _Tp >
```

[is\\_enum](#)

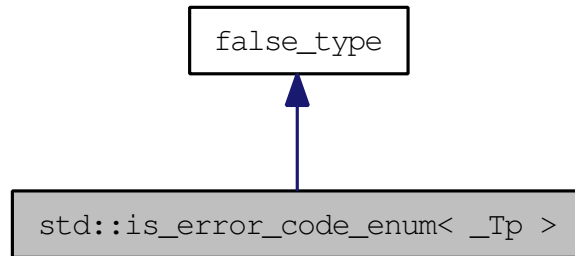
Definition at line 206 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.526 `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**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.526.1 Detailed Description

```
template<typename _Tp> struct std::is_error_code_enum< _Tp >
```

[is\\_error\\_code\\_enum](#)

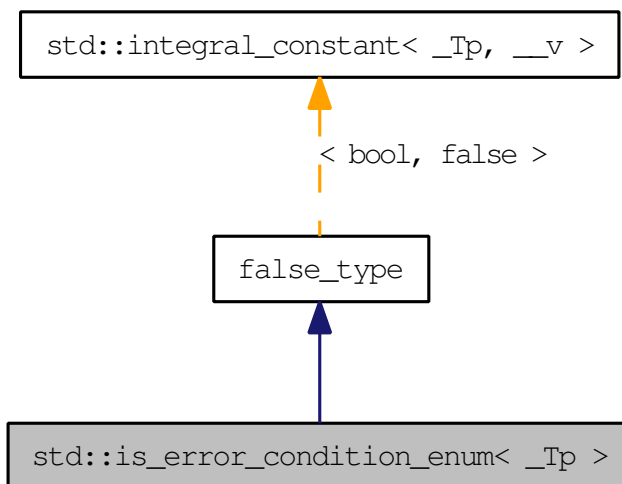
Definition at line 52 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.527 `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](#)`< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.527.1 Detailed Description

`template<typename _Tp> struct std::is_error_condition_enum< _Tp >`

[is\\_error\\_condition\\_enum](#)

Definition at line 56 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.528 `std::is_explicitly_convertible<_From, _To>` Struct Template Reference 2773

### 5.528 `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`

#### Static Public Attributes

- static const `bool value`

#### 5.528.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_explicitly_convertible<_From, _To>
```

[is\\_explicitly\\_convertible](#)

Definition at line 316 of file `type_traits`.

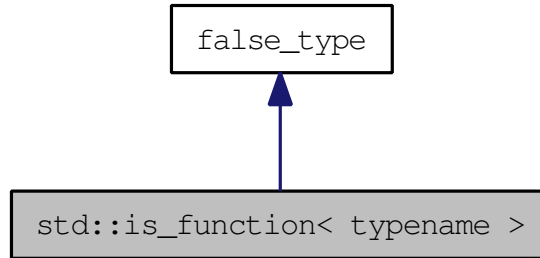
The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 5.530 `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**

### Static Public Attributes

- static const `_Tp` **value**

### 5.530.1 Detailed Description

`template<typename> struct std::is_function< typename >`

[is\\_function](#)

Definition at line 224 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.531 `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**

### Static Public Attributes

- static const bool **value**

#### 5.531.1 Detailed Description

`template<typename _Tp> struct std::is_fundamental< _Tp >`

[is\\_fundamental](#)

Definition at line 262 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)



## 5.532 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.532.1 Detailed Description

`template<typename _Tp> struct std::is_integral< _Tp >`

[is\\_integral](#)

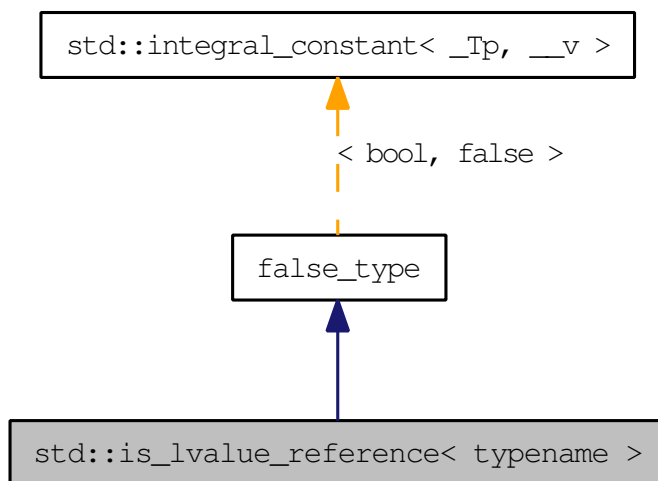
Definition at line 126 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.533 `std::is_lvalue_reference< typename >` Struct Template Reference

[is\\_lvalue\\_reference](#) Inheritance diagram for `std::is_lvalue_reference< typename >`:



### Public Types

- typedef [integral\\_constant](#)`< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.533.1 Detailed Description

`template<typename> struct std::is_lvalue_reference< typename >`

[is\\_lvalue\\_reference](#)

Definition at line 69 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.534 `std::is_member_function_pointer< _Tp >` Struct Template Reference 2779

### 5.534 `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**

#### Static Public Attributes

- static const `bool` **value**

#### 5.534.1 Detailed Description

`template<typename _Tp> struct std::is_member_function_pointer< _Tp >`

[is\\_member\\_function\\_pointer](#)

Definition at line 199 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.535 `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**

### Static Public Attributes

- static const bool **value**

#### 5.535.1 Detailed Description

`template<typename _Tp> struct std::is_member_object_pointer< _Tp >`

[is\\_member\\_object\\_pointer](#)

Definition at line 186 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.536 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**

### Static Public Attributes

- static const bool **value**

### 5.536.1 Detailed Description

**template<typename \_Tp> struct std::is\_object< \_Tp >**

[is\\_object](#)

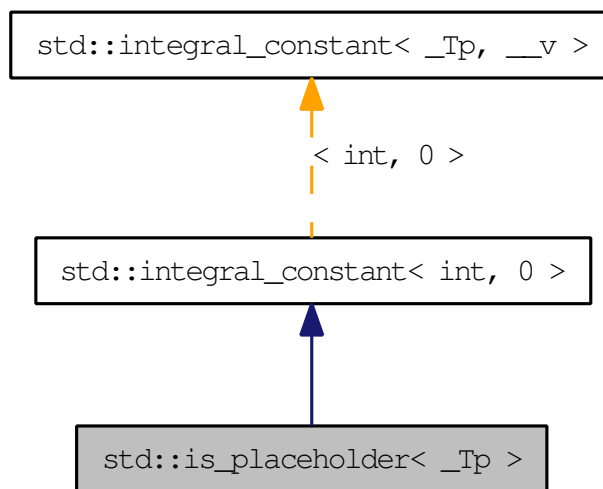
Definition at line 269 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.537 `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**

### Static Public Attributes

- static const `int` **value**

#### 5.537.1 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].

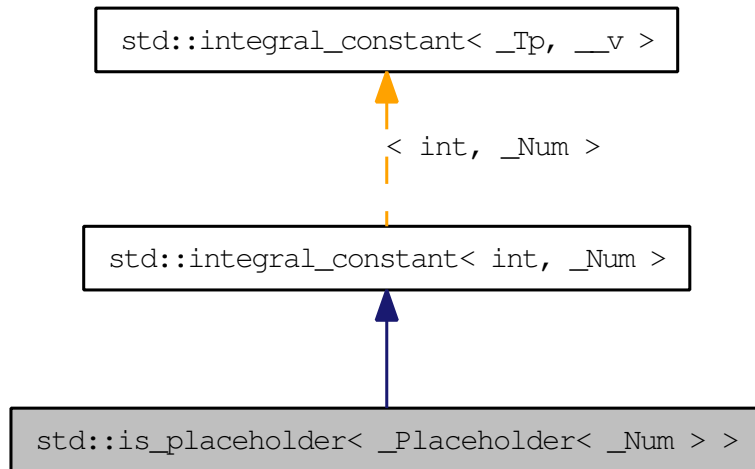
Definition at line 787 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.538 `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**

### Static Public Attributes

- static const `int` **value**

#### 5.538.1 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 844 of file `functional`.

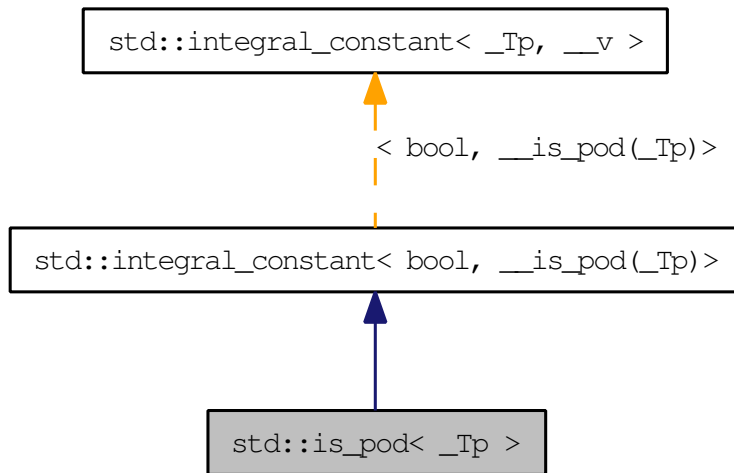
The documentation for this struct was generated from the following file:

- `functional`



## 5.539 std::is\_pod< \_Tp > Struct Template Reference

[is\\_pod](#) Inheritance diagram for std::is\_pod< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

### 5.539.1 Detailed Description

```
template<typename _Tp> struct std::is_pod< _Tp >
```

[is\\_pod](#)

Definition at line 191 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.540 `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**

### Static Public Attributes

- static const bool **value**

#### 5.540.1 Detailed Description

`template<typename _Tp> struct std::is_pointer< _Tp >`

[is\\_pointer](#)

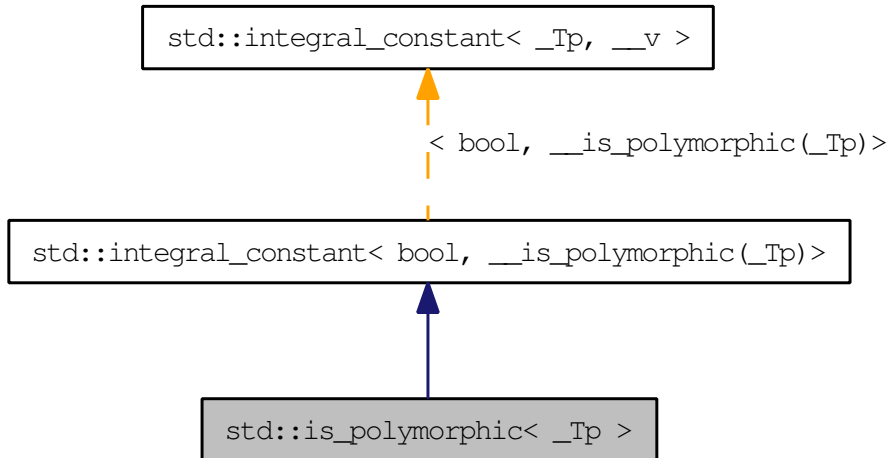
Definition at line 165 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.541 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**

### Static Public Attributes

- static const `bool` **value**

#### 5.541.1 Detailed Description

```
template<typename _Tp> struct std::is_polymorphic< _Tp >
```

[is\\_polymorphic](#)

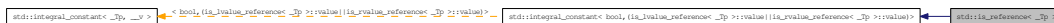
Definition at line 332 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.542 `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**

### Static Public Attributes

- static const bool **value**

#### 5.542.1 Detailed Description

`template<typename _Tp> struct std::is_reference< _Tp >`

[is\\_reference](#)

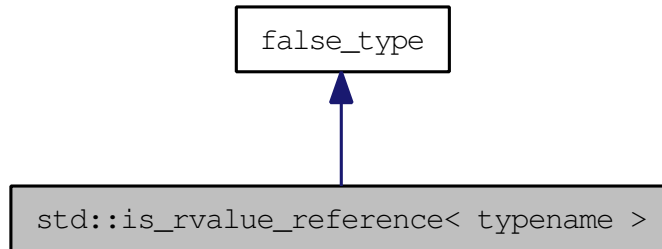
Definition at line 89 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.543 `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**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.543.1 Detailed Description

`template<typename> struct std::is_rvalue_reference< typename >`

[is\\_rvalue\\_reference](#)

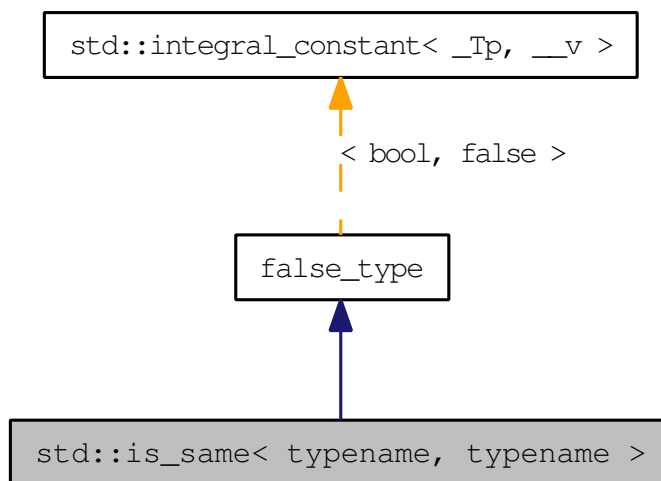
Definition at line 78 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.544 `std::is_same< typename, typename >` Struct Template Reference

[is\\_same](#) Inheritance diagram for `std::is_same< typename, typename >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.544.1 Detailed Description

`template<typename, typename> struct std::is_same< typename, typename >`

[is\\_same](#)

Definition at line 389 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- `tr1_impl/type_traits`

## 5.545 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**

### Static Public Attributes

- static const bool **value**

### 5.545.1 Detailed Description

`template<typename _Tp> struct std::is_scalar< _Tp >`

[is\\_scalar](#)

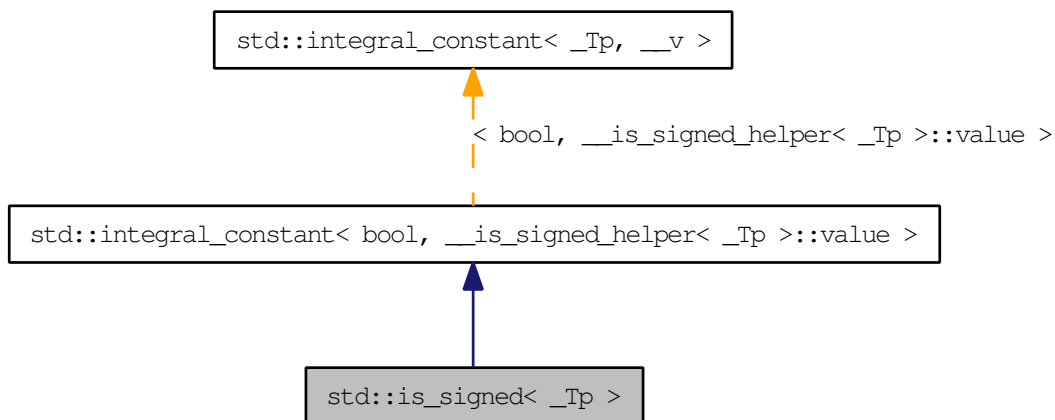
Definition at line 281 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.546 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.546.1 Detailed Description

```
template<typename _Tp> struct std::is_signed< _Tp >
```

[is\\_signed](#)

Definition at line 163 of file `type_traits`.

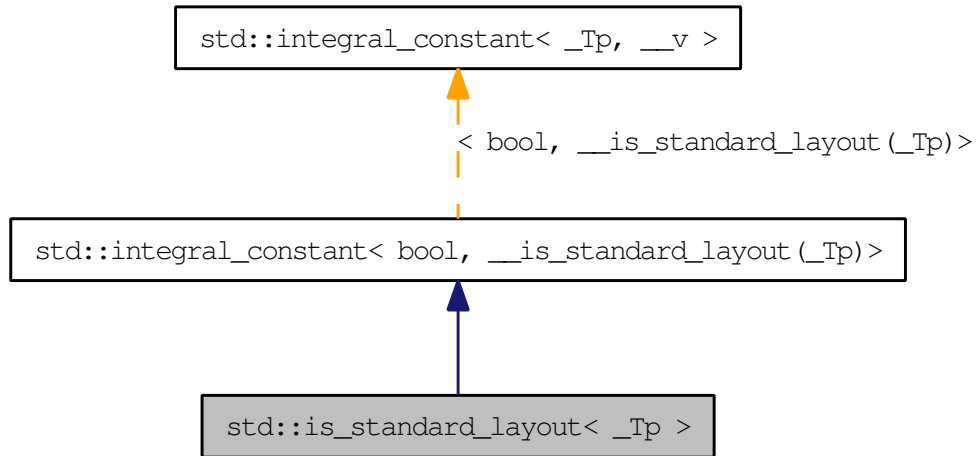
The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 5.547 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**

### Static Public Attributes

- static const `bool` **value**

#### 5.547.1 Detailed Description

```
template<typename _Tp> struct std::is_standard_layout< _Tp >
```

[is\\_standard\\_layout](#)

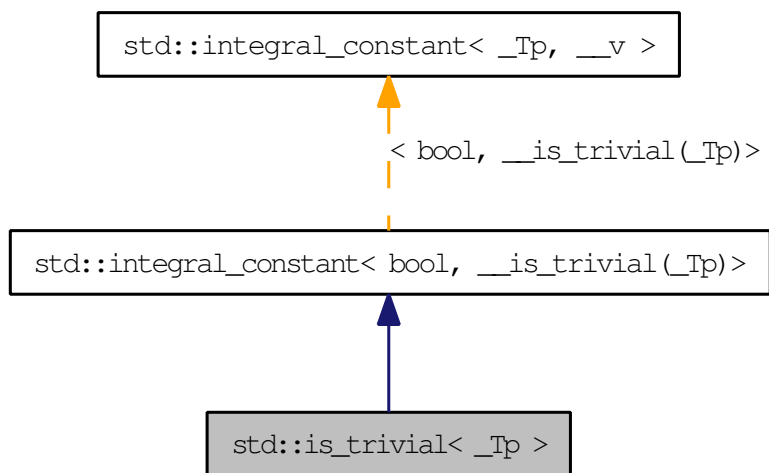
Definition at line 184 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.548 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.548.1 Detailed Description

```
template<typename _Tp> struct std::is_trivial< _Tp >
```

[is\\_trivial](#)

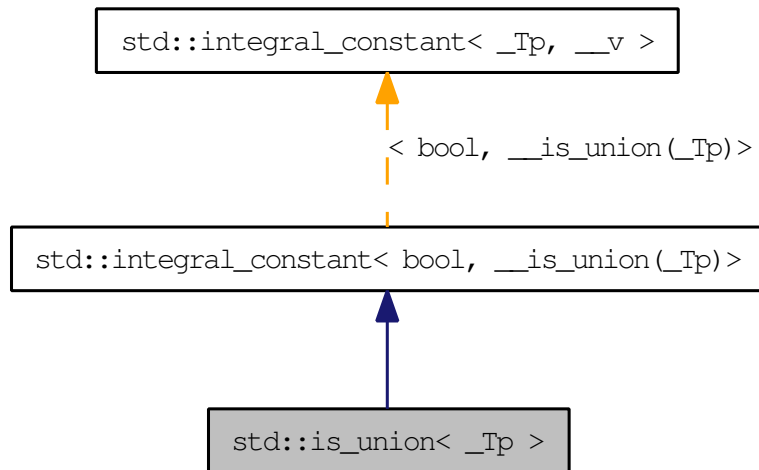
Definition at line 178 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.549 `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**

### Static Public Attributes

- static const `bool` **value**

### 5.549.1 Detailed Description

```
template<typename _Tp> struct std::is_union< _Tp >
```

[is\\_union](#)

Definition at line 212 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.550 `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**

### Static Public Attributes

- static const bool **value**

#### 5.550.1 Detailed Description

`template<typename _Tp> struct std::is_unsigned< _Tp >`

[is\\_unsigned](#)

Definition at line 169 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.551 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**

### Static Public Attributes

- static const `bool` **value**

#### 5.551.1 Detailed Description

`template<typename _Tp> struct std::is_void< _Tp >`

[is\\_void](#)

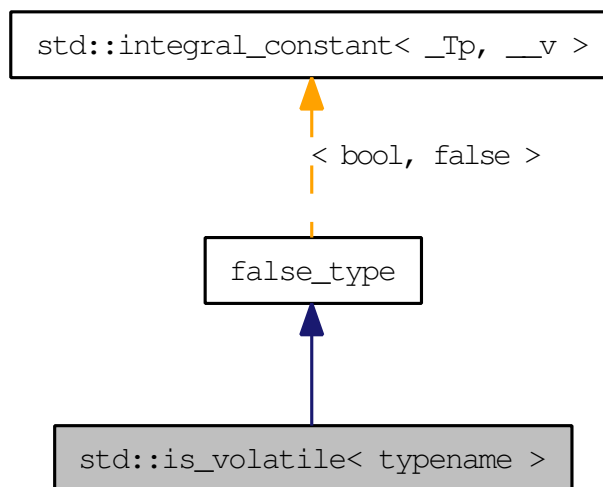
Definition at line 96 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.552 `std::is_volatile< typename >` Struct Template Reference

[is\\_volatile](#) Inheritance diagram for `std::is_volatile< typename >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Static Public Attributes

- static const bool **value**

#### 5.552.1 Detailed Description

`template<typename> struct std::is_volatile< typename >`

[is\\_volatile](#)

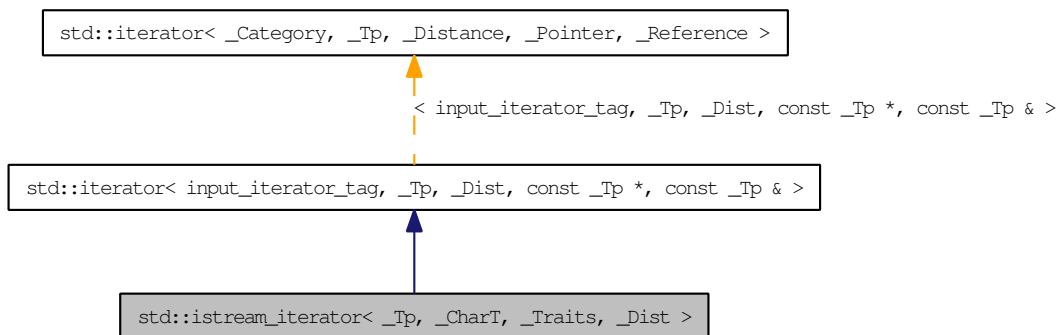
Definition at line 317 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.553 `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

- **istream\_iterator** (const `istream_iterator` &\_\_obj)
- **istream\_iterator** (`istream_type` &\_\_s)
- **istream\_iterator** ()
- `bool` **\_M\_equal** (const `istream_iterator` &\_\_x) const
- `const _Tp &` **operator\*** () const
- `istream_iterator` **operator++** (int)
- `istream_iterator` & **operator++** ()
- `const _Tp *` **operator->** () const

### 5.553.1 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 47 of file stream\_iterator.h.

### 5.553.2 Member Typedef Documentation

**5.553.2.1** `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 115 of file stl\_iterator\_base\_types.h.

**5.553.2.2** `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 111 of file stl\_iterator\_base\_types.h.

**5.553.2.3** `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 117 of file stl\_iterator\_base\_types.h.

**5.553.2.4** `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 119 of file stl\_iterator\_base\_types.h.

**5.553.2.5** `typedef _Tp std::iterator< input_iterator_tag , _Tp , _Dist , const _Tp * , const _Tp & >::value_type` [*inherited*]

The type "pointed to" by the iterator.



**5.553 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > Class Template Reference** **2801**

---

Definition at line 113 of file stl\_iterator\_base\_types.h.

### 5.553.3 Constructor & Destructor Documentation

**5.553.3.1** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline]`

Construct end of input stream [iterator](#).

Definition at line 62 of file stream\_iterator.h.

**5.553.3.2** `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 66 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.554 `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::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

- [istreambuf\\_iterator](#) (`streambuf_type * __s`) `throw ()`
- [istreambuf\\_iterator](#) (`istream_type & __s`) `throw ()`
- [istreambuf\\_iterator](#) () `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 >`)

## 5.554 `std::istreambuf_iterator<_CharT, _Traits>` Class Template Reference 2803

- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`  
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >`  
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`

### 5.554.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::istreambuf_iterator<`  
`_CharT, _Traits >`

Provides input `iterator` semantics for streambufs.

Definition at line 50 of file `streambuf_iterator.h`.

### 5.554.2 Member Typedef Documentation

**5.554.2.1** `template<typename _CharT, typename _Traits > typedef _CharT`  
`std::istreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 58 of file `streambuf_iterator.h`.

**5.554.2.2** `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 115 of file `stl_iterator_base_types.h`.

**5.554.2.3** `template<typename _CharT, typename _Traits > typedef`  
`_Traits::int_type std::istreambuf_iterator< _CharT, _Traits`  
`>::int_type`

Public typedefs.

Definition at line 60 of file `streambuf_iterator.h`.

**5.554.2.4** `template<typename _CharT, typename _Traits > typedef`  
`basic_istream< _CharT, _Traits> std::istreambuf_iterator< _CharT,`  
`_Traits >::istream_type`

Public typedefs.

Definition at line 62 of file streambuf\_iterator.h.

**5.554.2.5** `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 111 of file stl\_iterator\_base\_types.h.

**5.554.2.6** `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 117 of file stl\_iterator\_base\_types.h.

**5.554.2.7** `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 119 of file stl\_iterator\_base\_types.h.

**5.554.2.8** `template<typename _CharT , typename _Traits > typedef  
basic_streambuf< _CharT, _Traits> std::istreambuf_iterator<  
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 61 of file streambuf\_iterator.h.

**5.554.2.9** `template<typename _CharT , typename _Traits > typedef _Traits  
std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 59 of file streambuf\_iterator.h.

## **5.554 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

**5.554.2.10** `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 113 of file `stl_iterator_base_types.h`.

### **5.554.3 Constructor & Destructor Documentation**

**5.554.3.1** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator ()  
throw () [inline]`

Construct end of input stream [iterator](#).

Definition at line 96 of file `streambuf_iterator.h`.

**5.554.3.2** `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 100 of file `streambuf_iterator.h`.

**5.554.3.3** `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 104 of file `streambuf_iterator.h`.

### **5.554.4 Member Function Documentation**

**5.554.4.1** `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 160 of file `streambuf_iterator.h`.

**5.554.4.2** `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 111 of file [streambuf\\_iterator.h](#).

**5.554.4.3** `template<typename _CharT , typename _Traits >  
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits  
>::operator++ (int) [inline]`

Advance the [iterator](#). Calls [streambuf.sbumpc\(\)](#).

Definition at line 140 of file [streambuf\\_iterator.h](#).

References [std::basic\\_streambuf< \\_CharT, \\_Traits >::sbumpc\(\)](#).

**5.554.4.4** `template<typename _CharT , typename _Traits >  
istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits  
>::operator++ () [inline]`

Advance the [iterator](#). Calls [streambuf.sbumpc\(\)](#).

Definition at line 125 of file [streambuf\\_iterator.h](#).

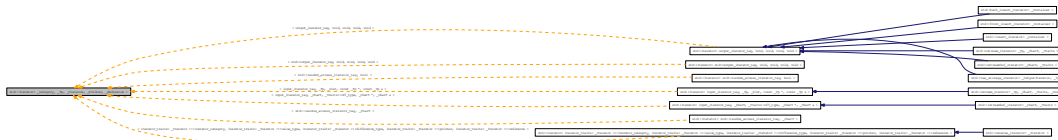
References [std::basic\\_streambuf< \\_CharT, \\_Traits >::sbumpc\(\)](#).

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

## 5.555 `std::iterator< _Category, _Tp, _Distance, _Pointer, _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](#)

### 5.555.1 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 108 of file `stl_iterator_base_types.h`.

### 5.555.2 Member Typedef Documentation

**5.555.2.1** `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 115 of file `stl_iterator_base_types.h`.

**5.555.2.2** `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 111 of file `stl_iterator_base_types.h`.

**5.555.2.3** `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 117 of file `stl_iterator_base_types.h`.

**5.555.2.4** `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 119 of file `stl_iterator_base_types.h`.

**5.555.2.5** `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 113 of file `stl_iterator_base_types.h`.

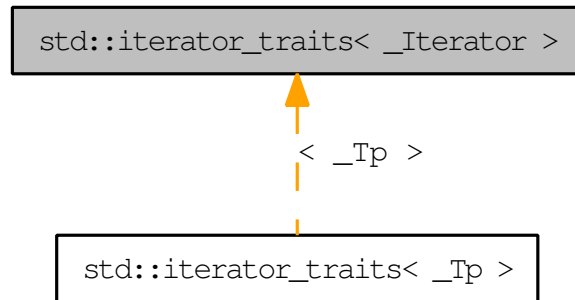
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)



## 5.556 std::iterator\_traits< \_Iterator > Struct Template Reference

Traits class for iterators. Inheritance diagram for std::iterator\_traits< \_Iterator >:



### Public Types

- typedef `_Iterator::difference_type` **difference\_type**
- typedef `_Iterator::iterator_category` **iterator\_category**
- typedef `_Iterator::pointer` **pointer**
- typedef `_Iterator::reference` **reference**
- typedef `_Iterator::value_type` **value\_type**

### 5.556.1 Detailed Description

```
template<typename _Iterator> struct std::iterator_traits< _Iterator >
```

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 131 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.557 `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**

### 5.557.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 142 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.558 `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**

### 5.558.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

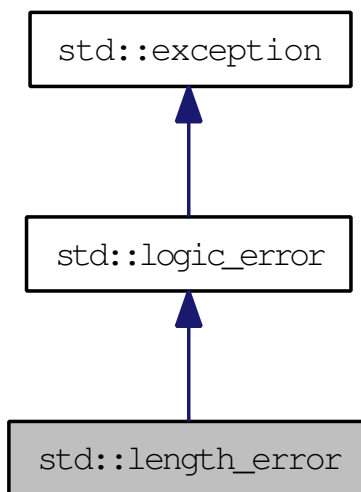
Definition at line 153 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.559 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 ()

#### 5.559.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a [basic\\_string](#) instance).

Definition at line 88 of file `stdexcept`.

#### 5.559.2 Member Function Documentation

##### 5.559.2.1 virtual const char\* std::length\_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](#).

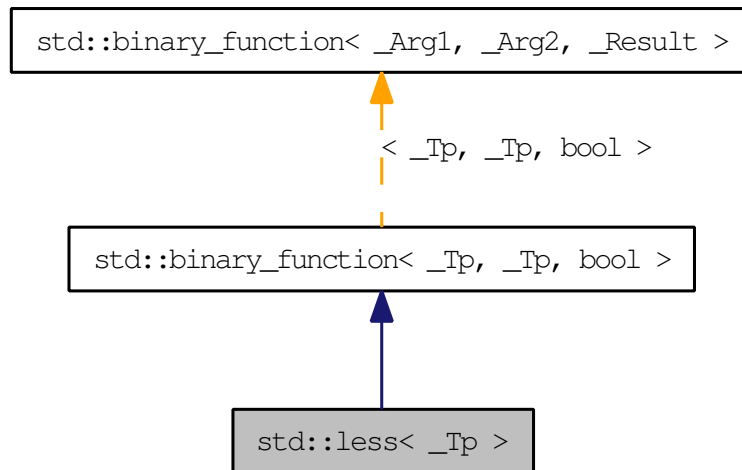
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.560 `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`

#### 5.560.1 Detailed Description

```
template<typename _Tp> struct std::less< _Tp >
```

One of the [comparison functors](#).

Definition at line 226 of file `stl_function.h`.

## 5.560.2 Member Typedef Documentation

**5.560.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.560.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.560.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

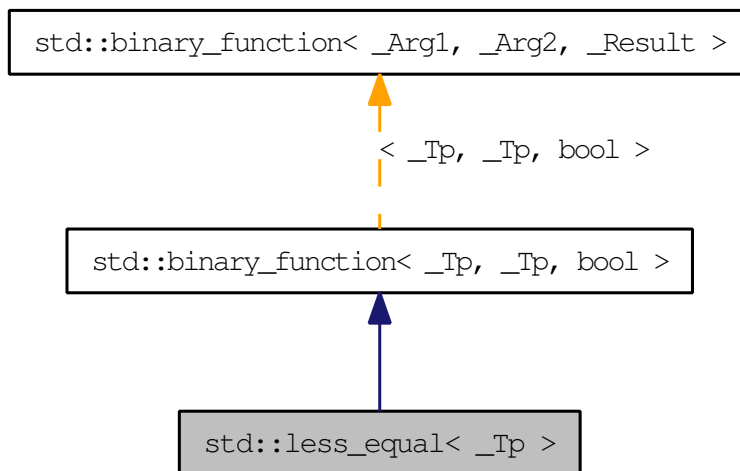
Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.561 `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`

#### 5.561.1 Detailed Description

```
template<typename _Tp> struct std::less_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 244 of file `stl_function.h`.



## 5.561.2 Member Typedef Documentation

**5.561.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.561.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.561.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.562 `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

- template<typename `_Sseq`, typename = typename `std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value >::type`> `linear_congruential_engine` (`_Sseq` &`_q`)
- `linear_congruential_engine` (`result_type` `__s`=`default_seed`)
- void `discard` (unsigned long long `__z`)
- `result_type` `max` () const
- `result_type` `min` () const
- `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 Attributes

- static const `result_type` `default_seed`
- static const `result_type` `increment`
- static const `result_type` `modulus`
- static const `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`)

## 5.562 `std::linear_congruential_engine<_UIntType, __a, __c, __m >` Class Template Reference 2819

---

- `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 > &)`

### 5.562.1 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 `(__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 157 of file `random.h`.

### 5.562.2 Member Typedef Documentation

**5.562.2.1** `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 166 of file `random.h`.

### 5.562.3 Constructor & Destructor Documentation

**5.562.3.1** `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 184 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

```
5.562.3.2 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 197 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

**5.562.4 Member Function Documentation**

```
5.562.4.1 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.

**Todo**

Look for a faster way to do discard.

Definition at line 247 of file random.h.

```
5.562.4.2 template<typename _UIntType, _UIntType __a, _UIntType __c,
_UIntType __m> result_type std::linear_congruential_engine<
_UIntType, __a, __c, __m >::max () const [inline]
```

Gets the largest possible value in the output range.

**Todo**

This should be `constexpr`.

Definition at line 238 of file `random.h`.

**5.562.4.3** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m >::min () const [inline]`

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.

**Todo**

This should be `constexpr`.

Definition at line 229 of file `random.h`.

**5.562.4.4** `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 257 of file `random.h`.

**5.562.4.5** `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) [inline]`

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 132 of file `random.tcc`.

References `std::lg()`, and `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

**5.562.4.6** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed (result_type __s = default_seed) [inline]`

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 116 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()`.

## 5.562.5 Friends And Related Function Documentation

**5.562.5.1** `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`.

5.562.5.2 `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 two objects are equal, false otherwise.

Definition at line 274 of file `random.h`.

5.562.5.3 `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`.

## 5.562.6 Member Data Documentation

**5.562.6.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m >::increment [inline, static]`

An increment.

Definition at line 171 of file random.h.

**5.562.6.2** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m >::modulus [inline, static]`

The [modulus](#).

Definition at line 173 of file random.h.

**5.562.6.3** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m >::multiplier [inline, static]`

The multiplier.

Definition at line 169 of file random.h.

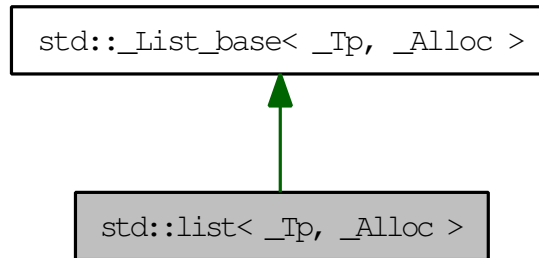
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)



## 5.563 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

- `template<typename _InputIterator >`  
`list` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const allocator_type &__a=allocator_type()`)
- `list` (`initializer_list< value_type >` \_\_l, `const allocator_type &__a=allocator_type()`)
- `list` (`list` &&\_\_x)
- `list` (`const list` &\_\_x)
- `list` (`size_type` \_\_n, `const value_type &__value=value_type()`, `const allocator_type &__a=allocator_type()`)

- [list](#) (const allocator\_type &\_\_a)
- [list](#) ()
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- 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)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- [const\\_iterator](#) [end](#) () const
- [iterator](#) [end](#) ()
- [iterator](#) [erase](#) ([iterator](#) \_\_first, [iterator](#) \_\_last)
- [iterator](#) [erase](#) ([iterator](#) \_\_position)
- const\_reference [front](#) () const
- reference [front](#) ()
- allocator\_type [get\\_allocator](#) () const
- template<typename \_InputIterator >  
void [insert](#) ([iterator](#) \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) ([iterator](#) \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- void [insert](#) ([iterator](#) \_\_p, initializer\_list< value\_type > \_\_l)
- [iterator](#) [insert](#) ([iterator](#) \_\_position, value\_type &&\_\_x)
- [iterator](#) [insert](#) ([iterator](#) \_\_position, const value\_type &\_\_x)
- size\_type [max\\_size](#) () const
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &\_\_x, \_StrictWeakOrdering \_\_comp)
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &&, \_StrictWeakOrdering)
- void [merge](#) ([list](#) &\_\_x)
- void [merge](#) ([list](#) && \_\_x)
- [list](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)

- `list & operator= (list &&__x)`
- `list & operator= (const list &__x)`
- `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)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `void remove (const _Tp &__value)`
- `template<typename _Predicate >`  
`void remove_if (_Predicate)`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void resize (size_type __new_size, value_type __x=value_type())`
- `void reverse ()`
- `size_type size () const`
- `template<typename _StrictWeakOrdering >`  
`void sort (_StrictWeakOrdering)`
- `void sort ()`
- `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, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x)`
- `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 _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __-`  
`false_type)`
- `template<typename _Integer >`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `void _M_check_equal_allocators (list &__x)`
- `void _M_clear ()`
- `template<typename... _Args>`  
`_Node * _M_create_node (_Args &&... __args)`
- `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 ()`
- `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 ()`
- `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`

### 5.563.1 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 417 of file `stl_list.h`.

## 5.563.2 Constructor & Destructor Documentation

**5.563.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc >::list () [inline]`

Default constructor creates no elements.

Definition at line 499 of file `stl_list.h`.

**5.563.2.2** `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 507 of file `stl_list.h`.

```
5.563.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (size_type _n, const value_type
& _value = value_type(), const allocator_type & _a =
allocator_type()) [inline, explicit]
```

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 519 of file `stl_list.h`.

```
5.563.2.4 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 531 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

```
5.563.2.5 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 543 of file `stl_list.h`.

```
5.563.2.6 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 554 of file `stl_list.h`.

```
5.563.2.7 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 571 of file `stl_list.h`.

### 5.563.3 Member Function Documentation

```
5.563.3.1 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 476 of file `stl_list.h`.

Referenced by `std::list< _Tp, _Alloc >::emplace()`, and `std::list< _Tp, _Alloc >::insert()`.

**5.563.3.2** `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 675 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::assign()`.

Referenced by `std::list< _Tp, _Alloc >::assign()`.

**5.563.3.3** `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 659 of file `stl_list.h`.

**5.563.3.4** `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 642 of file `stl_list.h`.

**5.563.3.5** `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 860 of file `stl_list.h`.

**5.563.3.6** `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 848 of file `stl_list.h`.

**5.563.3.7** `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 699 of file `stl_list.h`.

**5.563.3.8** `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 690 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc >::list()`, `std::list<_Tp, _Alloc >::merge()`, `std::list<_Tp, _Alloc >::operator=()`, `std::operator==(, std::list<_Tp, _Alloc >::remove()`, `std::list<_Tp, _Alloc >::remove_if()`, `std::list<_Tp, _Alloc >::resize()`, `std::list<_Tp, _Alloc >::sort()`, `std::list<_Tp, _Alloc >::splice()`, and `std::list<_Tp, _Alloc >::unique()`.

**5.563.3.9** `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 763 of file `stl_list.h`.

**5.563.3.10** `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 772 of file `stl_list.h`.

**5.563.3.11** `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 1127 of file `stl_list.h`.

**5.563.3.12** `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 781 of file `stl_list.h`.

**5.563.3.13** `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 790 of file `stl_list.h`.

**5.563.3.14** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> list< _Tp, _Alloc >::iterator list::emplace (iterator  
_position, _Args &&... _args) [inline]`

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 87 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::_M_create_node()`.

**5.563.3.15** `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 800 of file `stl_list.h`.

Referenced by `std::list< _Tp, _Alloc >::sort()`, and `std::list< _Tp, _Alloc >::splice()`.

**5.563.3.16** `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 717 of file `stl_list.h`.

**5.563.3.17** `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 708 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc>::list()`, `std::list<_Tp, _Alloc>::merge()`, `std::list<_Tp, _Alloc>::operator=()`, `std::operator==(())`, `std::list<_Tp, _Alloc>::remove()`, `std::list<_Tp, _Alloc>::remove_if()`, `std::list<_Tp, _Alloc>::resize()`, `std::list<_Tp, _Alloc>::splice()`, and `std::list<_Tp, _Alloc>::unique()`.

**5.563.3.18** `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 1093 of file `stl_list.h`.

**5.563.3.19** `template<typename _Tp , typename _Alloc > list<_Tp, _Alloc  
>::iterator list::erase (iterator __position) [inline]`

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 108 of file list.tcc.

Referenced by `std::list< _Tp, _Alloc >::operator=()`, and `std::list< _Tp, _Alloc >::resize()`.

**5.563.3.20** `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 840 of file stl\_list.h.

**5.563.3.21** `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 832 of file stl\_list.h.

**5.563.3.22** `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 681 of file stl\_list.h.

**5.563.3.23** `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 1049 of file `stl_list.h`.

```
5.563.3.24 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 1028 of file `stl_list.h`.

```
5.563.3.25 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 1011 of file `stl_list.h`.

References std::list< \_Tp, \_Alloc >::insert().

Referenced by std::list< \_Tp, \_Alloc >::insert().

**5.563.3.26** `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 994 of file stl\_list.h.

**5.563.3.27** `template<typename _Tp , typename _Alloc > list< _Tp, _Alloc  
>::iterator list::insert (iterator __position, const value_type & __x)  
[inline]`

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 98 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().

**5.563.3.28** `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 810 of file `stl_list.h`.

**5.563.3.29** `template<typename _Tp, typename _Alloc > template<typename  
_StrictWeakOrdering > void list::merge (list<_Tp, _Alloc > &&  
__x, _StrictWeakOrdering __comp) [inline]`

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 270 of file `list.tcc`.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

**5.563.3.30** `template<typename _Tp, typename _Alloc > void list::merge (list<  
_Tp, _Alloc > && __x) [inline]`

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 236 of file `list.tcc`.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

Referenced by `std::list<_Tp, _Alloc >::sort()`.



**5.563.3.31** `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 624 of file stl\_list.h.

**5.563.3.32** `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 607 of file stl\_list.h.

References std::swap().

**5.563.3.33** `template<typename _Tp, typename _Alloc > list< _Tp, _Alloc > &  
list::operator= (const list< _Tp, _Alloc > & __x) [inline]`

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 133 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()`.

**5.563.3.34** `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 946 of file `stl_list.h`.

**5.563.3.35** `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 906 of file `stl_list.h`.

**5.563.3.36** `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 920 of file `stl_list.h`.

**5.563.3.37** `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 879 of file stl\_list.h.

**5.563.3.38** `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 735 of file stl\_list.h.

**5.563.3.39** `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 726 of file stl\_list.h.

**5.563.3.40** `template<typename _Tp, typename _Alloc > void list::remove  
(const _Tp & __value) [inline]`

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 187 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

**5.563.3.41** `template<typename _Tp, typename _Alloc > template<typename  
_Predicate > void list::remove_if (_Predicate __pred) [inline]`

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 340 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

**5.563.3.42** `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 753 of file stl\_list.h.

**5.563.3.43** `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 744 of file stl\_list.h.

**5.563.3.44** `template<typename _Tp, typename _Alloc > void list::resize  
(size_type __new_size, value_type __x = value_type())  
[inline]`

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 118 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().

**5.563.3.45** `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 1347 of file stl\_list.h.

**5.563.3.46** `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 805 of file stl\_list.h.

References `__gnu_cxx::distance()`.

**5.563.3.47** `template<typename _Tp , typename _Alloc > template<typename  
_StrictWeakOrdering > void list::sort (_StrictWeakOrdering  
_comp) [inline]`

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 379 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().

**5.563.3.48** `template<typename _Tp, typename _Alloc > void list::sort ()`  
**[inline]**

Sort the elements. Sorts the elements of this [list](#) in NlogN time. Equivalent elements remain in [list](#) order.

Definition at line 302 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()`.

**5.563.3.49** `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 1213 of file stl\_list.h.

**5.563.3.50** `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 1177 of file stl\_list.h.

**5.563.3.51** `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 1147 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 >::sort()`.

**5.563.3.52** `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 1110 of file `stl_list.h`.

References `std::swap()`.

Referenced by `std::list< _Tp, _Alloc >::sort()`, and `std::swap()`.

**5.563.3.53** `template<typename _Tp, typename _Alloc > template<typename  
_BinaryPredicate > void list::unique (_BinaryPredicate  
__binary_pred) [inline]`

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 358 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

#### **5.563.3.54** `template<typename _Tp , typename _Alloc > void list::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 215 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](#)



## 5.564 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

- `template<typename _Facet >`  
`locale` (const `locale` &\_\_other, `_Facet` \*\_\_f)
- `locale` (const `locale` &\_\_base, const `locale` &\_\_add, `category` \_\_cat)
- `locale` (const `locale` &\_\_base, const char \*\_\_s, `category` \_\_cat)
- `locale` (const char \*\_\_s)
- `locale` (const `locale` &\_\_other) throw ()
- `locale` () throw ()
- `~locale` () throw ()
- `template<typename _Facet >`  
`locale combine` (const `locale` &\_\_other) const
- `string name` () const
- `bool operator!=` (const `locale` &\_\_other) const throw ()
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator()` (const `basic_string`< `_CharT`, `_Traits`, `_Alloc` > &\_\_s1, const `basic_string`< `_CharT`, `_Traits`, `_Alloc` > &\_\_s2) const

- `template<typename _Char, typename _Traits, typename _Alloc >`  
`bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const`  
`basic_string< _Char, _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 all`
- `static const category collate`
- `static const category ctype`
- `static const category messages`
- `static const category monetary`
- `static const category none`
- `static const category numeric`
- `static const category time`

## 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 &)`

### 5.564.1 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 62 of file `locale_classes.h`.

## 5.564.2 Member Typedef Documentation

### 5.564.2.1 typedef int std::locale::category

Definition of [locale::category](#).

Definition at line 67 of file locale\_classes.h.

## 5.564.3 Constructor & Destructor Documentation

### 5.564.3.1 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\(\)](#).

### 5.564.3.2 std::locale::locale (const locale & \_\_other) throw ()

Copy constructor. Constructs a copy of *other*.

#### Parameters:

*other* The [locale](#) to copy.

### 5.564.3.3 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](#).

### 5.564.3.4 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](#).

**5.564.3.5 `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*.

**5.564.3.6 `template<typename _Facet > std::locale::locale (const locale & __other, _Facet * __f) [inline]`**

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 43 of file `locale_classes.tcc`.

**5.564.3.7 `std::locale::~~locale () throw ()`**

Locale destructor.

## 5.564.4 Member Function Documentation

### 5.564.4.1 static const locale& std::locale::classic () [static]

Return reference to the C [locale](#).

### 5.564.4.2 template<typename \_Facet > locale std::locale::combine (const locale & \_\_other) const [inline]

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 61 of file `locale_classes.tcc`.

References `locale()`.

### 5.564.4.3 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](#).

**5.564.4.4** `string std::locale::name () const`

Return [locale](#) name.

**Returns:**

Locale name or "\*" if unnamed.

**5.564.4.5** `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 234 of file locale\_classes.h.

**5.564.4.6** `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 [inline]`

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.

**5.564.4.7 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](#).

**5.564.4.8 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.

**5.564.5 Member Data Documentation****5.564.5.1 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 105 of file `locale_classes.h`.

**5.564.5.2 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 101 of file `locale_classes.h`.

**5.564.5.3 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 99 of file `locale_classes.h`.

**5.564.5.4 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 104 of file `locale_classes.h`.

**5.564.5.5 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 103 of file `locale_classes.h`.

**5.564.5.6 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 98 of file `locale_classes.h`.

**5.564.5.7 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 100 of file locale\_classes.h.

#### 5.564.5.8 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 102 of file locale\_classes.h.

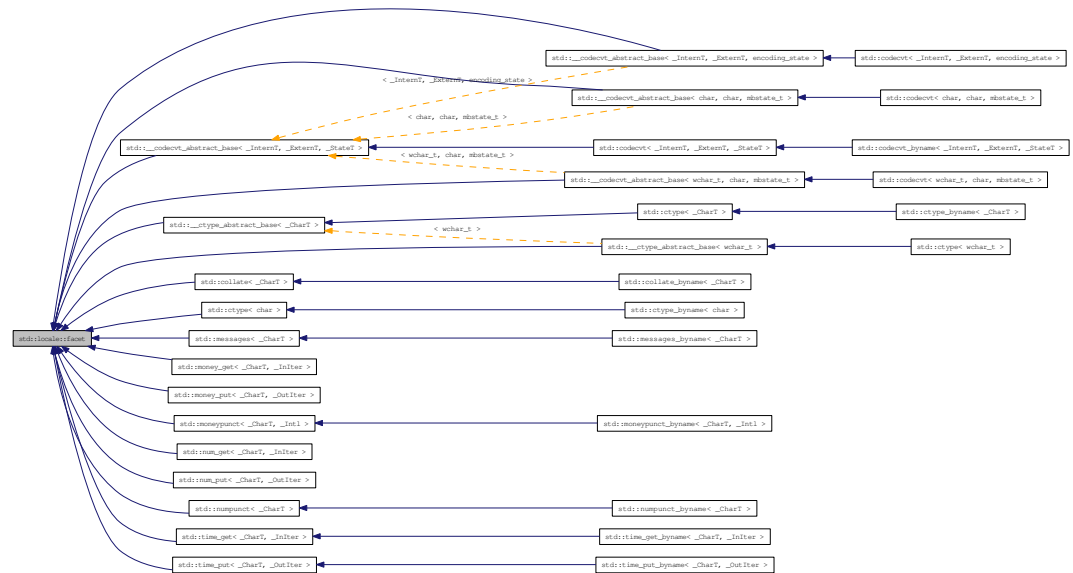
The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 5.565 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 ()`
- `__attribute__((__const__))` `static const char *_S_get_c_name() throw ()`

### 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 __c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale`
- class `locale::_Impl`

### 5.565.1 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 337 of file `locale_classes.h`.

### 5.565.2 Constructor & Destructor Documentation

#### 5.565.2.1 `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 369 of file `locale_classes.h`.

#### 5.565.2.2 `virtual std::locale::facet::~~facet () [protected, virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.566 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 &)`

### 5.566.1 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 432 of file [locale\\_classes.h](#).

### 5.566.2 Constructor & Destructor Documentation

#### 5.566.2.1 std::locale::id::id () [inline]

Constructor.

Definition at line 463 of file [locale\\_classes.h](#).

### 5.566.3 Friends And Related Function Documentation

#### 5.566.3.1 `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.

#### 5.566.3.2 `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](#)

## 5.567 `std::lock_guard<_Mutex>` Class Template Reference

Scoped lock idiom.

### Public Types

- typedef `_Mutex` `mutex_type`

### Public Member Functions

- `lock_guard` (const `lock_guard` &)
- `lock_guard` (`mutex_type` &\_\_m, `adopt_lock_t`)
- `lock_guard` (`mutex_type` &\_\_m)
- `lock_guard` & `operator=` (const `lock_guard` &)

#### 5.567.1 Detailed Description

`template<typename _Mutex> class std::lock_guard<_Mutex>`

Scoped lock idiom.

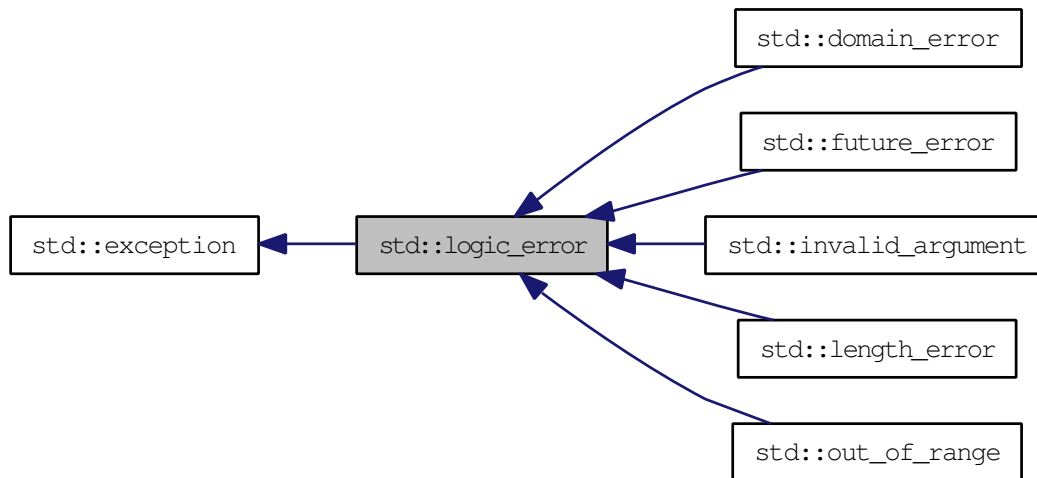
Definition at line 393 of file `mutex`.

The documentation for this class was generated from the following file:

- `mutex`

## 5.568 `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 ()

#### 5.568.1 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 53 of file `stdexcept`.

#### 5.568.2 Constructor & Destructor Documentation

##### 5.568.2.1 `std::logic_error::logic_error` (const `string` & \_\_arg) [`explicit`]

Takes a character string describing the error.

### 5.568.3 Member Function Documentation

#### 5.568.3.1 `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](#).

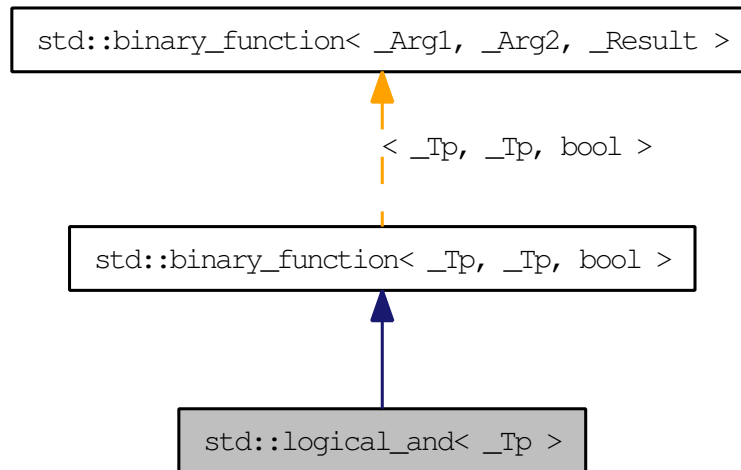
The documentation for this class was generated from the following file:

- [stdexcept](#)



## 5.569 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`

#### 5.569.1 Detailed Description

```
template<typename _Tp> struct std::logical_and< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 263 of file `stl_function.h`.

## 5.569.2 Member Typedef Documentation

**5.569.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.569.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.569.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

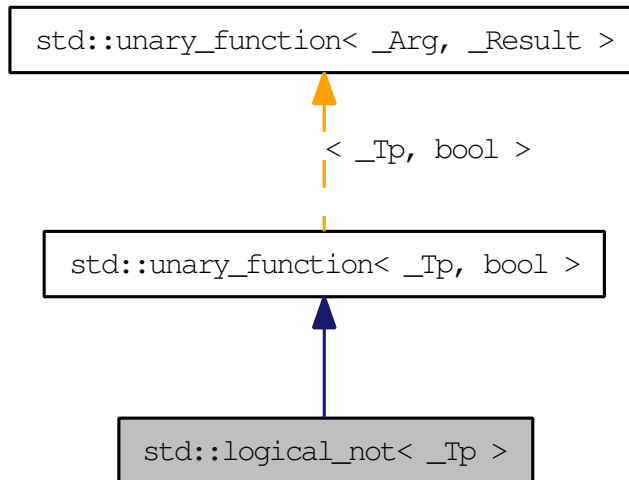
Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.570 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`

#### 5.570.1 Detailed Description

```
template<typename _Tp> struct std::logical_not< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 281 of file `stl_function.h`.

## 5.570.2 Member Typedef Documentation

### 5.570.2.1 `typedef _Tp std::unary_function< _Tp , bool >::argument_type` [*inherited*]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.570.2.2 `typedef bool std::unary_function< _Tp , bool >::result_type` [*inherited*]

`result_type` is the return type

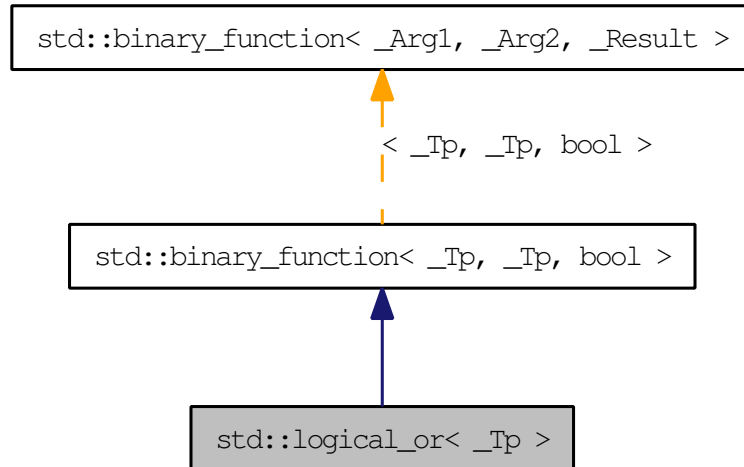
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.571 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`

#### 5.571.1 Detailed Description

```
template<typename _Tp> struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 272 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]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.571.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.571.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.572 `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` (const `param_type` &\_\_p)
- `lognormal_distribution` (`_RealType` \_\_m=`_RealType`(0), `_RealType` \_\_s=`_RealType`(1))
- `_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`, typename `_CharT`, typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & `operator`>> (`std::basic_istream`< `_CharT`, `_Traits` > &, `std::lognormal_distribution`< `_RealType1` > &)

### 5.572.1 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 1970 of file random.h.

### 5.572.2 Member Typedef Documentation

**5.572.2.1 `template<typename _RealType = double> typedef _RealType std::lognormal_distribution<_RealType>::result_type`**

The type of the range of the distribution.

Definition at line 1977 of file random.h.

### 5.572.3 Member Function Documentation

**5.572.3.1 `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 2057 of file random.h.

**5.572.3.2 `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 2050 of file random.h.

**5.572.3.3 `template<typename _RealType = double> void std::lognormal_distribution<_RealType>::param (const param_type & __param) [inline]`**

Sets the parameter [set](#) of the distribution.



## 5.572 std::lognormal\_distribution< \_RealType > Class Template Reference 2873

### Parameters:

*\_\_param* The new parameter [set](#) of the distribution.

Definition at line 2043 of file random.h.

**5.572.3.4** `template<typename _RealType = double> param_type  
std::lognormal_distribution< _RealType >::param () const  
[inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 2035 of file random.h.

**5.572.3.5** `template<typename _RealType = double> void  
std::lognormal_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2017 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

## 5.572.4 Friends And Related Function Documentation

**5.572.4.1** `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`.

### 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.

**5.572.4.2** `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits > &, std::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](#)

## **5.573 `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

### **5.573.1 Detailed Description**

**template<typename `_RealType` = `double`> struct `std::lognormal_distribution<_RealType>::param_type`**

Parameter type.

Definition at line 1979 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.574 `std::make_signed< _Tp >` Struct Template Reference

[make\\_signed](#)

### Public Types

- `typedef __make_signed_selector< _Tp >::__type type`

#### 5.574.1 Detailed Description

`template<typename _Tp> struct std::make_signed< _Tp >`

[make\\_signed](#)

Definition at line 587 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.575 `std::make_unsigned< _Tp >` Struct Template Reference

[make\\_unsigned](#)

### Public Types

- `typedef __make_unsigned_selector< _Tp >::__type type`

#### 5.575.1 Detailed Description

```
template<typename _Tp> struct std::make_unsigned< _Tp >
```

[make\\_unsigned](#)

Definition at line 510 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.576 `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

- `template<typename _InputIterator >`  
`map` (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp`,  
`const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
`map` (`_InputIterator __first`, `_InputIterator __last`)
- `map` (`initializer_list< value_type > __l`, `const _Compare &__c=_Compare()`,  
`const allocator_type &__a=allocator_type()`)
- `map` (`map &&__x`)
- `map` (`const map &__x`)
- `map` (`const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `map` ()
- `const mapped_type &` `at` (`const key_type &__k`) `const`
- `mapped_type &` `at` (`const key_type &__k`)
- `const_iterator` `begin` () `const`
- `iterator` `begin` ()

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2879

- 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
- bool [empty](#) () const
- const\_iterator [end](#) () const
- iterator [end](#) ()
- 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](#) (iterator \_\_first, iterator \_\_last)
- size\_type [erase](#) (const key\_type &\_\_x)
- iterator [erase](#) (iterator \_\_position)
- const\_iterator [find](#) (const key\_type &\_\_x) const
- iterator [find](#) (const key\_type &\_\_x)
- allocator\_type [get\\_allocator](#) () const
- template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator [insert](#) (iterator \_\_position, const value\_type &\_\_x)
- void [insert](#) (std::initializer\_list< value\_type > \_\_list)
- std::pair< iterator, bool > [insert](#) (const value\_type &\_\_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
- map & operator= (initializer\_list< value\_type > \_\_l)
- map & operator= (map &&\_\_x)
- map & operator= (const map &\_\_x)
- mapped\_type & operator[] (const 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)
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- 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 > &)`

### 5.576.1 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 86 of file `stl_map.h`.

### 5.576.2 Constructor & Destructor Documentation

**5.576.2.1** `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 150 of file `stl_map.h`.



## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2881

**5.576.2.2** `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.

Definition at line 159 of file stl\_map.h.

**5.576.2.3** `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 170 of file stl\_map.h.

**5.576.2.4** `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 181 of file stl\_map.h.

```

5.576.2.5 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 *N*log*N* otherwise (where *N* is *l.size()*).

Definition at line 195 of file `stl_map.h`.

```

5.576.2.6 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 *N*log*N* otherwise (where *N* is `distance(first,last)`).

Definition at line 212 of file `stl_map.h`.

```

5.576.2.7 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.

## 5.576 `std::map<_Key, _Tp, _Compare, _Alloc >` Class Template Reference 2883

### Parameters:

- first* An input [iterator](#).
- last* An input [iterator](#).
- 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 228 of file `stl_map.h`.

### 5.576.3 Member Function Documentation

**5.576.3.1** `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 465 of file `stl_map.h`.

**5.576.3.2** `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 316 of file `stl_map.h`.

**5.576.3.3** `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 307 of file `stl_map.h`.

**5.576.3.4** `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 380 of file `stl_map.h`.

**5.576.3.5** `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 389 of file `stl_map.h`.

**5.576.3.6** `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 663 of file `stl_map.h`.

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2885

**5.576.3.7** `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 723 of file `stl_map.h`.

**5.576.3.8** `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 398 of file `stl_map.h`.

**5.576.3.9** `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 407 of file `stl_map.h`.

**5.576.3.10** `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 416 of file stl\_map.h.

**5.576.3.11** `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 334 of file stl\_map.h.

**5.576.3.12** `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 325 of file stl\_map.h.

**5.576.3.13** `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.

**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).

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2887

This function probably only makes sense for multimaps.

Definition at line 811 of file stl\_map.h.

```
5.576.3.14 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 792 of file stl\_map.h.

```
5.576.3.15 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 (iterator __first, 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 622 of file `stl_map.h`.

**5.576.3.16** `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 603 of file `stl_map.h`.

**5.576.3.17** `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(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 573 of file `stl_map.h`.



## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2889

**5.576.3.18** `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 711 of file `stl_map.h`.

**5.576.3.19** `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 696 of file `stl_map.h`.

**5.576.3.20** `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 297 of file stl\_map.h.

```
5.576.3.21 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 553 of file stl\_map.h.

```
5.576.3.22 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 (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 540 of file stl\_map.h.

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2891

**5.576.3.23** `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 512 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

**5.576.3.24** `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 500 of file `stl_map.h`.

**5.576.3.25** `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 672 of file `stl_map.h`.

**5.576.3.26** `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\(\)](#).

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 753 of file `stl_map.h`.

**5.576.3.27** `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 738 of file `stl_map.h`.

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2893

**5.576.3.28** `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 426 of file stl\_map.h.

**5.576.3.29** `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 287 of file stl\_map.h.

**5.576.3.30** `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.

### 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 266 of file stl\_map.h.

References `std::swap()`.

```

5.576.3.31 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 251 of file `stl_map.h`.

```

5.576.3.32 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 443 of file `stl_map.h`.

## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2895

**5.576.3.33** `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 352 of file `stl_map.h`.

**5.576.3.34** `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 343 of file `stl_map.h`.

**5.576.3.35** `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 370 of file `stl_map.h`.

**5.576.3.36** `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 361 of file `stl_map.h`.

**5.576.3.37** `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 421 of file `stl_map.h`.

**5.576.3.38** `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 653 of file `stl_map.h`.

Referenced by `std::swap()`.

**5.576.3.39** `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 773 of file `stl_map.h`.



## 5.576 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2897

**5.576.3.40** `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 763 of file [stl\\_map.h](#).

**5.576.3.41** `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 680 of file [stl\\_map.h](#).

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 5.577 `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
- `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
- `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
- `template<class _Ex > void operator=` (const `_Expr< _Ex, _Tp >` &\_\_e) const
- `template<class _Dom > void operator=` (const `_Expr< _Dom, _Tp >` &) const
- `void operator=` (const `_Tp` &) const
- `void operator=` (const `valarray< _Tp >` &) const
- `mask_array` & `operator=` (const `mask_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

## Friends

- class `valarray<_Tp>`

### 5.577.1 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 61 of file `mask_array.h`.

The documentation for this class was generated from the following file:

- `mask_array.h`

## 5.578 `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** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, `std::forward_iterator_tag`)
- void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, `std::input_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\_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** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- void **\_M\_insert\_aux** (iterator \_\_position, \_Args &&... \_\_args)
- void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)

## 5.578 `std::match_results<_Bi_iter, _Allocator >` Class Template Reference 2901

- void `_M_insert_dispatch` (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_-type)
- void `_M_range_check` (size\_type \_\_n) const
- void `_M_range_initialize` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void `_M_range_initialize` (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_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` (initializer\_list< value\_type > \_\_l)
- void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (size\_type \_\_n, const value\_type &\_\_val)
- const\_reference `at` (size\_type \_\_n) const
- reference `at` (size\_type \_\_n)
- const\_reference `back` () const
- reference `back` ()
- iterator `begin` ()
- size\_type `capacity` () const
- void `clear` ()
- const\_reverse\_iterator `crbegin` () const
- const\_reverse\_iterator `crend` () const
- const\_pointer `data` () const
- pointer `data` ()
- iterator `emplace` (iterator \_\_position, \_Args &&...\_\_args)
- void `emplace_back` (\_Args &&...\_\_args)
- iterator `end` ()
- iterator `erase` (iterator \_\_first, iterator \_\_last)
- iterator `erase` (iterator \_\_position)
- const\_reference `front` () const
- reference `front` ()
- allocator\_type `get_allocator` () const
- void `insert` (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void `insert` (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- void `insert` (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- iterator `insert` (iterator \_\_position, value\_type &&\_\_x)
- iterator `insert` (iterator \_\_position, const value\_type &\_\_x)
- size\_type `max_size` () const
- const\_reference `operator[]` (size\_type \_\_n) const
- reference `operator[]` (size\_type \_\_n)
- void `pop_back` ()
- void `push_back` (value\_type &&\_\_x)

- void `push_back` (const `value_type` &\_\_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 \_\_new\_size, `value_type` \_\_x=`value_type`())
- void `shrink_to_fit` ()
- void `swap` (`vector` &\_\_x)

### Private Attributes

- `_Vector_impl_M_impl`

## 10.? Public Types

- typedef `_Allocator` `allocator_type`
- typedef `iterator_traits`< `_Bi_iter` >::`value_type` `char_type`
- typedef `_Base_type`::`const_iterator` `const_iterator`
- typedef `_Allocator`::`const_reference` `const_reference`
- typedef `iterator_traits`< `_Bi_iter` >::`difference_type` `difference_type`
- typedef `const_iterator` `iterator`
- typedef `const_reference` `reference`
- typedef `_Allocator`::`size_type` `size_type`
- typedef `basic_string`< `char_type` > `string_type`
- typedef `sub_match`< `_Bi_iter` > `value_type`

### 10.1 Construction, Copying, and Destruction

- `match_results` (const `match_results` &\_\_rhs)
- `match_results` (const `_Allocator` &\_\_a=`_Allocator`())
- `~match_results` ()
- `match_results` & `operator=` (const `match_results` &\_\_rhs)

### 10.3 Element Access

- `const_iterator` `begin` () const
- `const_iterator` `cbegin` () const
- `const_iterator` `end` () const
- `const_iterator` `end` () const

## 5.578 `std::match_results<_Bi_iter, _Allocator >` Class Template Reference 2903

- difference\_type `length` (size\_type \_\_sub=0) const
- const\_reference `operator[]` (size\_type \_\_sub) const
- difference\_type `position` (size\_type \_\_sub=0) const
- const\_reference `prefix` () const
- string\_type `str` (size\_type \_\_sub=0) const
- const\_reference `suffix` () const

### 10.2 Size

- bool `empty` () const
- size\_type `size` () 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.

- string\_type `format` (const string\_type &\_\_fmt, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::format\_default) const
- 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

### 10.6 Swap

- void `swap` (match\_results &\_\_that)

#### 5.578.1 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 1765 of file `tr1_impl/regex`.

## 5.578.2 Constructor & Destructor Documentation

**5.578.2.1** `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 1801 of file `tr1_impl/regex`.

**5.578.2.2** `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 1808 of file `tr1_impl/regex`.

**5.578.2.3** `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 1827 of file `tr1_impl/regex`.



### 5.578.3 Member Function Documentation

**5.578.3.1** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter> >> const_iterator std::match_results<_Bi_iter, _Allocator >::begin () const [inline]`

Gets an [iterator](#) to the start of the `sub_match` collection.

Reimplemented from [std::vector< std::\\_GLIBCXX\\_TR1 sub\\_match<\\_Bi\\_iter >, \\_Allocator >](#).

Definition at line 1949 of file `tr1_impl/regex`.

**5.578.3.2** `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::\\_GLIBCXX\\_TR1 sub\\_match<\\_Bi\\_iter >, \\_Allocator >](#).

Definition at line 1957 of file `tr1_impl/regex`.

**5.578.3.3** `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::\\_GLIBCXX\\_TR1 sub\\_match<\\_Bi\\_iter >, \\_Allocator >](#).

Definition at line 1973 of file `tr1_impl/regex`.

**5.578.3.4** `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::_GLIBCXX_TR1 sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1860 of file `tr1_impl/regex`.

```
5.578.3.5 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::_GLIBCXX_TR1 sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1965 of file `tr1_impl/regex`.

```
5.578.3.6 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.

```
5.578.3.7 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.

```
5.578.3.8 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.

## 5.578 std::match\_results< \_Bi\_iter, \_Allocator > Class Template Reference 2907

### 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 1878 of file tr1\_impl/regex.

```
5.578.3.9 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 1817 of file tr1\_impl/regex.

```
5.578.3.10 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 1920 of file tr1\_impl/regex.

```
5.578.3.11 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 function returns the offset from the beginning of the target sequence to the beginning of the match.

Definition at line 1892 of file `tr1_impl/regex`.

**5.578.3.12** `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 1931 of file `tr1_impl/regex`.

Referenced by `std::match_results<_Bi_iter>::position()`.

**5.578.3.13** `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::_GLIBCXX_TR1 sub_match<_Bi_iter>, _Allocator >`.

Definition at line 1847 of file `tr1_impl/regex`.

Referenced by `std::match_results<_Bi_iter>::empty()`.

**5.578.3.14** `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.

## 5.578 std::match\_results< \_Bi\_iter, \_Allocator > Class Template Reference 2909

### 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 1906 of file `tr1_impl/regex`.

Referenced by `std::match_results< _Bi_iter >::length()`.

```
5.578.3.15 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 1942 of file `tr1_impl/regex`.

```
5.578.3.16 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 2031 of file `tr1_impl/regex`.

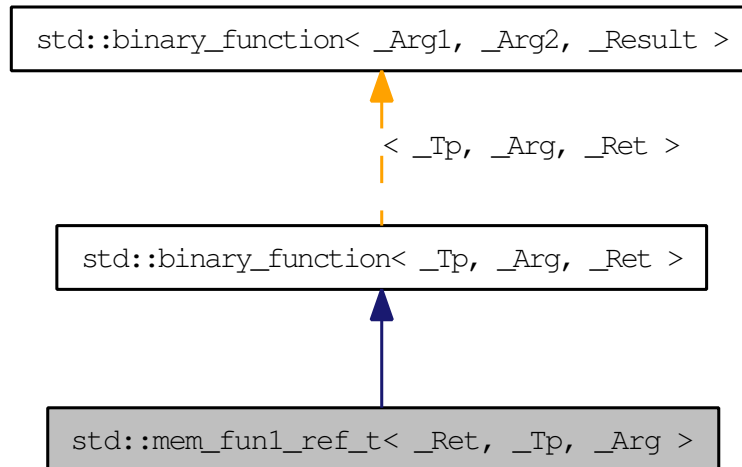
Referenced by `std::match_results< _Bi_iter >::operator=()`, `std::swap()`, and `std::match_results< _Bi_iter >::swap()`.

The documentation for this class was generated from the following file:

- [tr1\\_impl/regex](#)

## 5.579 `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`

#### 5.579.1 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 632 of file `stl_function.h`.

## 5.579.2 Member Typedef Documentation

**5.579.2.1** `typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.579.2.2** `typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.579.2.3** `typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]`

the type of the second argument

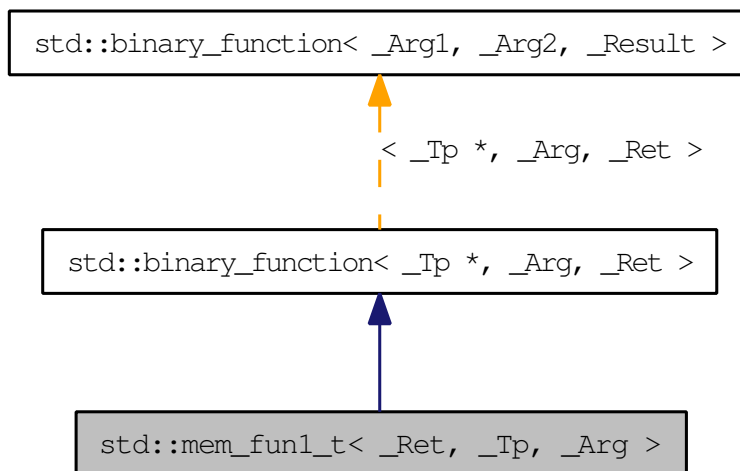
Definition at line 117 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.580 `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`

#### 5.580.1 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 596 of file `stl_function.h`.



## 5.580.2 Member Typedef Documentation

**5.580.2.1** `typedef _Tp * std::binary_function< _Tp *, _Arg , _Ret >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.580.2.2** `typedef _Ret std::binary_function< _Tp *, _Arg , _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.580.2.3** `typedef _Arg std::binary_function< _Tp *, _Arg , _Ret >::second_argument_type [inherited]`

the type of the second argument

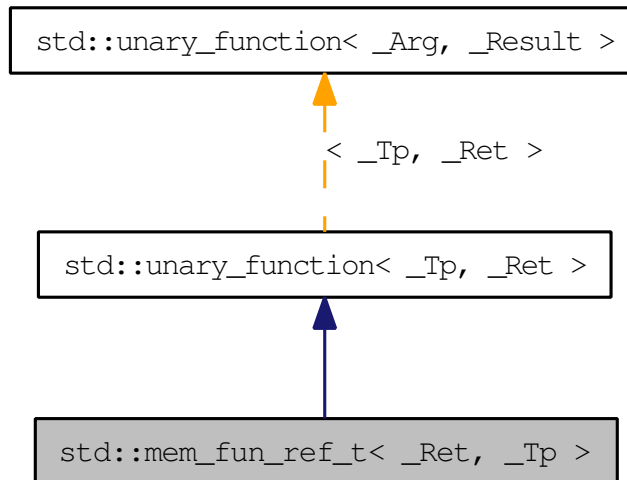
Definition at line 117 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.581 `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`

#### 5.581.1 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 560 of file `stl_function.h`.

## 5.581.2 Member Typedef Documentation

### 5.581.2.1 `typedef _Tp std::unary_function<_Tp, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.581.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [`inherited`]

`result_type` is the return type

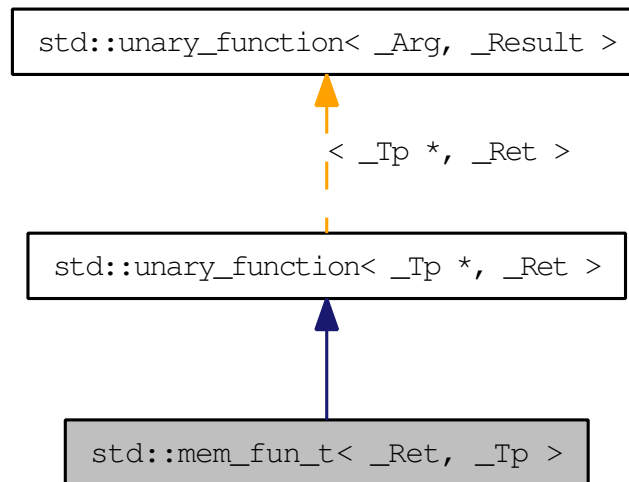
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.582 `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

#### 5.582.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::mem_fun_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 524 of file `stl_function.h`.

## 5.582.2 Member Typedef Documentation

### 5.582.2.1 `typedef _Tp * std::unary_function<_Tp *, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.582.2.2 `typedef _Ret std::unary_function<_Tp *, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

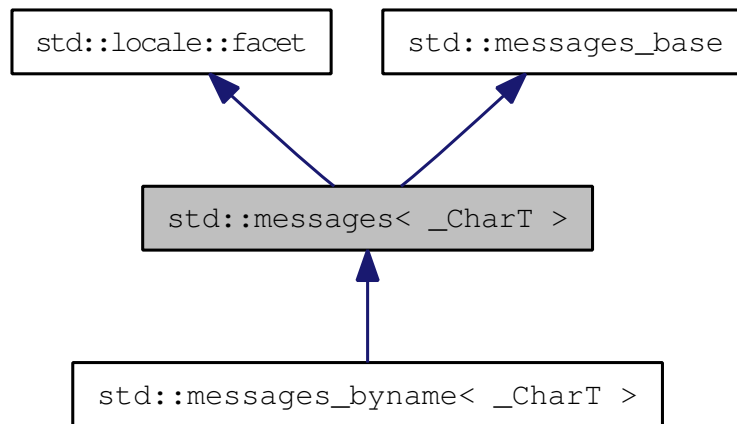
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.583 `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](#) (`__c_locale __cloc`, `const char *__s`, `size_t __refs=0`)
- [messages](#) (`size_t __refs=0`)
- void **close** (`catalog __c`) const
- template<>  
[wstring do\\_get](#) (`catalog`, `int`, `int`, `const wstring &`) const
- template<>  
[string do\\_get](#) (`catalog`, `int`, `int`, `const string &`) 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

- virtual [~messages](#) ()
- [\\_\\_attribute\\_\\_](#) ((\_\_const\_\_)) static const char \*\_S\_get\_c\_name() throw ()
- [string\\_type](#) **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const [string\\_type](#) &\_\_msg) const
- virtual void **do\_close** (catalog) 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 [\\_\\_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](#)

### 5.583.1 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 1689 of file `locale_facets_nonio.h`.

### 5.583.2 Member Typedef Documentation

**5.583.2.1 template<typename \_CharT > typedef \_CharT std::messages< \_CharT >::char\_type**

Public typedefs.

Reimplemented in [std::messages\\_byname< \\_CharT >](#).

Definition at line 1695 of file `locale_facets_nonio.h`.

**5.583.2.2 template<typename \_CharT > typedef basic\_string<\_CharT> std::messages< \_CharT >::string\_type**

Public typedefs.

Reimplemented in [std::messages\\_byname< \\_CharT >](#).

Definition at line 1696 of file `locale_facets_nonio.h`.

### 5.583.3 Constructor & Destructor Documentation

**5.583.3.1 template<typename \_CharT > std::messages< \_CharT >::messages (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 43 of file messages\_members.h.

**5.583.3.2** `template<typename _CharT> std::messages<_CharT>::messages  
(__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* Refcount to pass to the base class.

Definition at line 49 of file messages\_members.h.

**5.583.3.3** `template<typename _CharT> std::messages<_CharT  
>::~messages() [inline, protected, virtual]`

Destructor.

Definition at line 78 of file messages\_members.h.

## 5.583.4 Member Data Documentation

**5.583.4.1** `template<typename _CharT> locale::id std::messages<_CharT  
>::id [inline, static]`

Numpunct facet id.

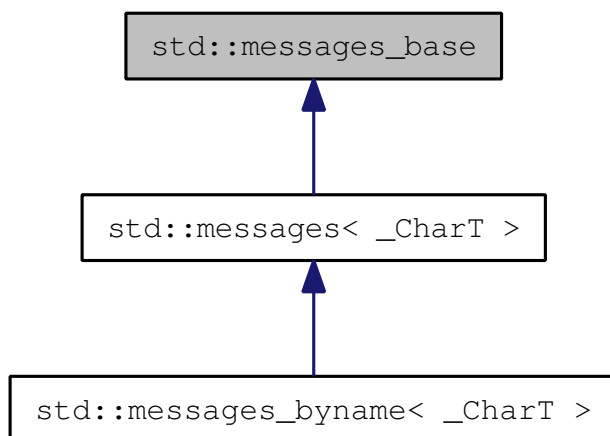
Definition at line 1707 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.584 std::messages\_base Struct Reference

Messages facet base class providing catalog typedef. Inheritance diagram for std::messages\_base:



### Public Types

- typedef int **catalog**

### 5.584.1 Detailed Description

Messages facet base class providing catalog typedef.

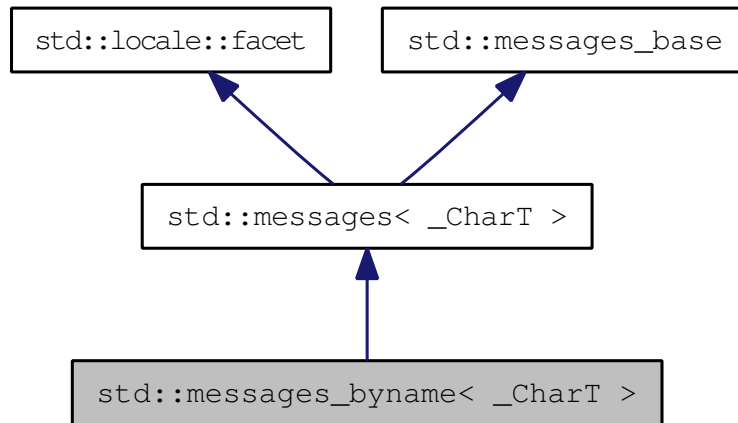
Definition at line 1662 of file `locale_facets_nonio.h`.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.585 `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`
- `template<>`  
`wstring do_get` (`catalog`, `int`, `int`, `const wstring &`) `const`
- `template<>`  
`string do_get` (`catalog`, `int`, `int`, `const string &`) `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

- `__attribute__((__const__))` static const char \*\_S\_get\_c\_name() throw ()
- [string\\_type](#) \_M\_convert\_from\_char (char \*) const
- char \*\_M\_convert\_to\_char (const [string\\_type](#) &\_\_msg) const
- virtual void do\_close (catalog) 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 `__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](#)

### 5.585.1 Detailed Description

`template<typename _CharT> class std::messages_byname< _CharT >`

class [messages\\_byname](#) [22.2.7.2].

Definition at line 1906 of file `locale_facets_nonio.h`.

## 5.585.2 Member Typedef Documentation

### 5.585.2.1 `template<typename _CharT> typedef _CharT std::messages_byname<_CharT>::char_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT>](#).

Definition at line 1909 of file `locale_facets_nonio.h`.

### 5.585.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::messages_byname<_CharT>::string_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT>](#).

Definition at line 1910 of file `locale_facets_nonio.h`.

## 5.585.3 Member Data Documentation

### 5.585.3.1 `template<typename _CharT> locale::id std::messages<_CharT> >::id [inline, static, inherited]`

Numpunct facet id.

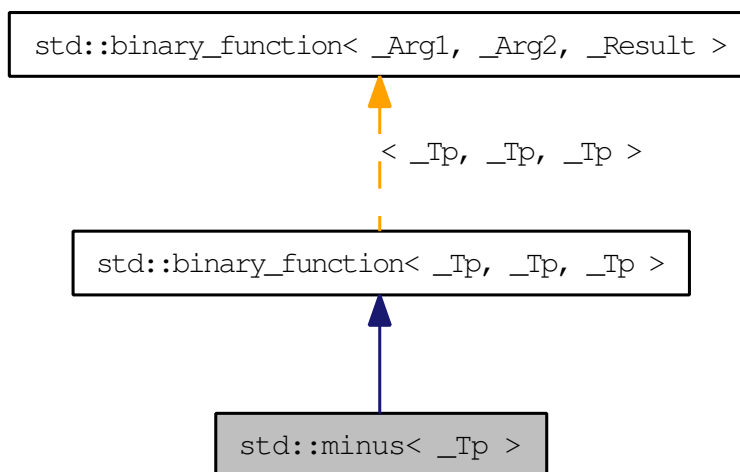
Definition at line 1707 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.586 `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`

#### 5.586.1 Detailed Description

```
template<typename _Tp> struct std::minus<_Tp>
```

One of the [math functors](#).

Definition at line 144 of file `stl_function.h`.

## 5.586.2 Member Typedef Documentation

**5.586.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.586.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.586.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]`

the type of the second argument

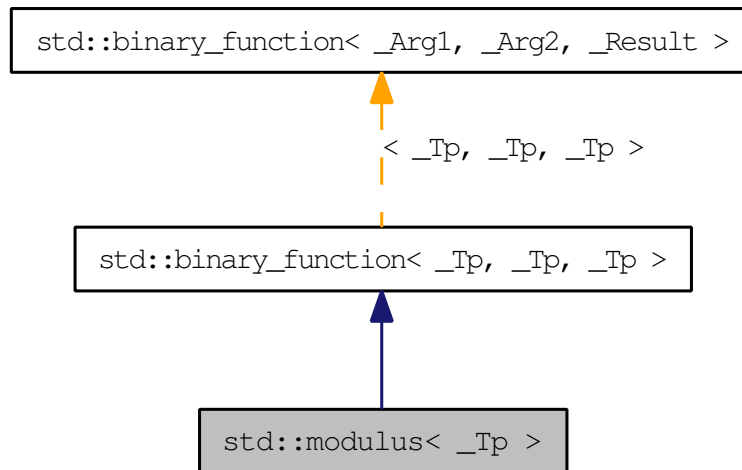
Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.587 `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

#### 5.587.1 Detailed Description

```
template<typename _Tp> struct std::modulus<_Tp>
```

One of the [math functors](#).

Definition at line 171 of file `stl_function.h`.



## 5.587.2 Member Typedef Documentation

**5.587.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.587.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.587.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

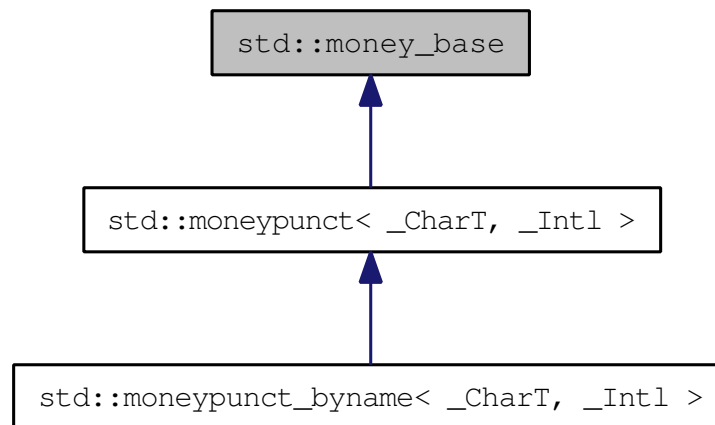
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.588 `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` }

### Public Member Functions

- `__attribute__ ((__const__)) static pattern _S_construct_pattern(char __precedes`
- `char char __posn throw ()`

### Public Attributes

- `char __space`

## Static Public Attributes

- static const char \* `_S_atoms`
- static const pattern `_S_default_pattern`

### 5.588.1 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 835 of file `locale_facets_nonio.h`.

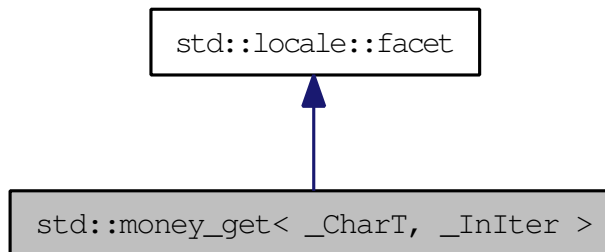
The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.589 `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](#) ()
- [\\_\\_attribute\\_\\_](#) ((`__const__`)) static const char \*\_S\_get\_c\_name() throw ()

- `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 `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc`, `const char *__s`)

## Friends

- class `locale::_Impl`

### 5.589.1 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 1364 of file `locale_facets_nonio.h`.

### 5.589.2 Member Typedef Documentation

**5.589.2.1** `template<typename _CharT, typename _InIter> typedef _CharT std::money_get<_CharT, _InIter>::char_type`

Public typedefs.

Definition at line 1370 of file `locale_facets_nonio.h`.

**5.589.2.2** `template<typename _CharT, typename _InIter > typedef _InIter  
std::money_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1371 of file locale\_facets\_nonio.h.

**5.589.2.3** `template<typename _CharT, typename _InIter > typedef  
basic_string<_CharT> std::money_get< _CharT, _InIter  
>::string_type`

Public typedefs.

Definition at line 1372 of file locale\_facets\_nonio.h.

### 5.589.3 Constructor & Destructor Documentation

**5.589.3.1** `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 1386 of file locale\_facets\_nonio.h.

**5.589.3.2** `template<typename _CharT, typename _InIter > virtual  
std::money_get< _CharT, _InIter >::~~money_get () [inline,  
protected, virtual]`

Destructor.

Definition at line 1454 of file locale\_facets\_nonio.h.

## 5.589.4 Member Function Documentation

**5.589.4.1** `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, string_type & __digits) const [inline, 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 376 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()`.

**5.589.4.2** `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, long  
double & __units) const [inline, 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 363 of file locale\_facets\_nonio.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`.

Referenced by `std::money_get< _CharT, _InIter >::get()`.

**5.589.4.3** `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<money_punct<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 1447 of file `locale_facets_nonio.h`.

References `std::money_get<_CharT, _InIter >::do_get()`.

**5.589.4.4** `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 [money\\_punct](#) and [ctype](#) facets retrieved from `io.getloc()`, and returns the result in *units* as an integral value `money_punct::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<money_punct<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 1416 of file `locale_facets_nonio.h`.

References `std::money_get<_CharT, _InIter >::do_get()`.

### 5.589.5 Member Data Documentation

**5.589.5.1** `template<typename _CharT, typename _InIter > locale::id  
std::money_get<_CharT, _InIter >::id [inline, static]`

Numpunct facet id.

Definition at line 1376 of file `locale_facets_nonio.h`.

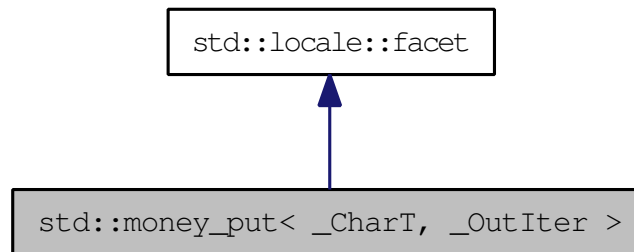
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.590 `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](#) ()
- `__attribute__ ((__const__))` static `const char *_S_get_c_name()` `throw ()`

- `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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.590.1 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 1515 of file `locale_facets_nonio.h`.

### 5.590.2 Member Typedef Documentation

**5.590.2.1** `template<typename _CharT, typename _OutIter> typedef _CharT std::money_put<_CharT, _OutIter>::char_type`

Public typedefs.

Definition at line 1520 of file `locale_facets_nonio.h`.

**5.590.2.2** `template<typename _CharT, typename _OutIter > typedef _OutIter  
std::money_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 1521 of file locale\_facets\_nonio.h.

**5.590.2.3** `template<typename _CharT, typename _OutIter > typedef  
basic_string<_CharT> std::money_put< _CharT, _OutIter  
>::string_type`

Public typedefs.

Definition at line 1522 of file locale\_facets\_nonio.h.

### 5.590.3 Constructor & Destructor Documentation

**5.590.3.1** `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 1536 of file locale\_facets\_nonio.h.

**5.590.3.2** `template<typename _CharT, typename _OutIter > virtual  
std::money_put< _CharT, _OutIter >::~~money_put () [inline,  
protected, virtual]`

Destructor.

Definition at line 1586 of file locale\_facets\_nonio.h.

## 5.590.4 Member Function Documentation

**5.590.4.1** `template<typename _CharT, typename _OutIter > _OutIter  
std::money_put<_CharT, _OutIter >::do_put (iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, const string_type &  
__digits) const [inline, 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 606 of file `locale_facets_nonio.tcc`.

**5.590.4.2** `template<typename _CharT, typename _OutIter > _OutIter  
std::money_put<_CharT, _OutIter >::do_put (iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, long double __units) const  
[inline, 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<money\_punct<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 568 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().

**5.590.4.3** `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 [money\\_punct](#) 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<money\_punct<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 1579 of file locale\_facets\_nonio.h.

References std::money\_put< \_CharT, \_OutIter >::do\_put().

**5.590.4.4** `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 `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 1556 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _OutIter >::do_put()`.

## 5.590.5 Member Data Documentation

**5.590.5.1** `template<typename _CharT, typename _OutIter > locale::id  
std::money_put<_CharT, _OutIter >::id [inline, static]`

Numpunct facet id.

Definition at line 1526 of file `locale_facets_nonio.h`.

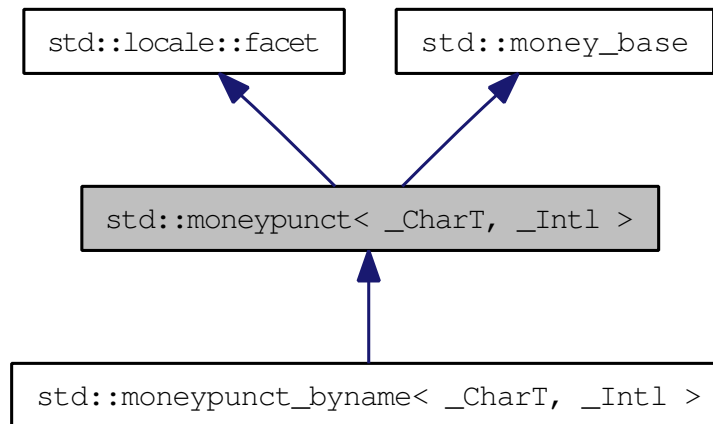
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.591 `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` }

### Public Member Functions

- `moneypunct` (`__c_locale __cloc`, `const char *__s`, `size_t __refs=0`)
- `moneypunct` (`__cache_type *__cache`, `size_t __refs=0`)
- `moneypunct` (`size_t __refs=0`)
- `__attribute__` (`((__const__))`) static `pattern _S_construct_pattern(char __precedes`
- `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 *)`
- `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
- `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`
- `char char __posn throw ()`

## Public Attributes

- `char __space`

## 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 ()`
- `__attribute__((__const__)) static const char *_S_get_c_name() throw ()`
- `void _M_initialize_moneypunct (__c_locale __cloc=NULL, const char *__name=NULL)`
- `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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`
- typedef `_CharT char_type`
- typedef `basic_string<_CharT> string_type`
- pattern `neg_format () const`
- pattern `pos_format () const`

### 5.591.1 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 929 of file `locale_facets_nonio.h`.

### 5.591.2 Member Typedef Documentation

**5.591.2.1** `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 935 of file `locale_facets_nonio.h`.

```
5.591.2.2 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 936 of file `locale_facets_nonio.h`.

### 5.591.3 Constructor & Destructor Documentation

```
5.591.3.1 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 958 of file `locale_facets_nonio.h`.

```
5.591.3.2 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 970 of file `locale_facets_nonio.h`.

```
5.591.3.3 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 985 of file locale\_facets\_nonio.h.

**5.591.3.4** `template<typename _CharT, bool _Intl> virtual std::moneypunct<_CharT, _Intl >::~~moneypunct () [protected, virtual]`

Destructor.

## 5.591.4 Member Function Documentation

**5.591.4.1** `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 1055 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

**5.591.4.2** `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 999 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

**5.591.4.3** `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 1201 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::curr_symbol()`.

**5.591.4.4** `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 1163 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

**5.591.4.5** `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 1241 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::frac_digits()`.

**5.591.4.6** `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 1188 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::grouping()`.

**5.591.4.7** `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 1269 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::neg_format()`.

**5.591.4.8** `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 1227 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::negative_sign()`.

**5.591.4.9** `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 1255 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::pos_format()`.

**5.591.4.10** `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 1214 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::positive_sign()`.

**5.591.4.11** `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 1175 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

**5.591.4.12** `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 1105 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.



**5.591.4.13** `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 interpret 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 1042 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

**5.591.4.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct< _CharT, _Intl >::neg_format () const [inline]`

Public typedefs.

Definition at line 1145 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

**5.591.4.15** `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 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 1089 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_negative_sign()`.

#### 5.591.4.16 `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 1141 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_pos_format()`.

#### 5.591.4.17 `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 1072 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_positive_sign()`.

**5.591.4.18** `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 `moneypunct<char_type>::do_thousands_sep()`.

**Returns:**

`char_type` representing a thousands separator.

Definition at line 1012 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

## 5.591.5 Member Data Documentation

**5.591.5.1** `template<typename _CharT, bool _Intl> locale::id  
std::moneypunct<_CharT, _Intl >::id [inline, static]`

Numpunct facet id.

Definition at line 948 of file `locale_facets_nonio.h`.

**5.591.5.2** `template<typename _CharT, bool _Intl> const bool  
std::moneypunct<_CharT, _Intl >::intl [inline, static]`

This value is provided by the standard, but no reason for its existence.

Reimplemented in `std::moneypunct_byname<_CharT, _Intl >`.

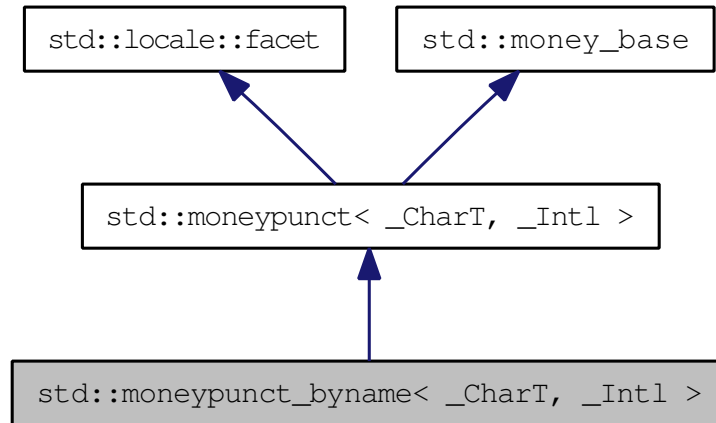
Definition at line 946 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.592 `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`)
- `__attribute__((const))` static `pattern` `_S_construct_pattern(char __precedes`
- `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 *)`
  - `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
  - `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`
  - `char char __posn throw ()`
- 
- pattern `neg_format () const`
  - pattern `pos_format () const`

## Public Attributes

- `char __space`

## 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

- `__attribute__((__const__)) static const char *_S_get_c_name() throw ()`
- `void _M_initialize_moneypunct (__c_locale __cloc=NULL, const char *__name=NULL)`
- 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`

## 5.592 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference2959

### 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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

### 5.592.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct_byname< _CharT, _Intl >`

class [moneypunct\\_byname](#) [22.2.6.4].

Definition at line 1318 of file `locale_facets_nonio.h`.

### 5.592.2 Member Typedef Documentation

**5.592.2.1** `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 1321 of file `locale_facets_nonio.h`.

**5.592.2.2** `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 1322 of file `locale_facets_nonio.h`.

### 5.592.3 Member Function Documentation

**5.592.3.1** `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 `moneypunct<char_type>::do_curr_symbol()`.

**Returns:**

*string\_type* representing a currency symbol.

Definition at line 1055 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

**5.592.3.2** `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 `moneypunct<char_type>::do_decimal_point()`.

**Returns:**

*char\_type* representing a `decimal` point.

Definition at line 999 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

**5.592.3.3** `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.



## 5.592 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference 2961

Definition at line 1201 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::curr\_symbol().

**5.592.3.4** `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 1163 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::decimal\_point().

**5.592.3.5** `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.

Definition at line 1241 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::frac\_digits().

**5.592.3.6** `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 1188 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::grouping()`.

**5.592.3.7** `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 1269 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::neg_format()`.

**5.592.3.8** `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 1227 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::negative_sign()`.

## 5.592 std::moneypunct\_byname< \_CharT, \_Intl > Class Template Reference2963

**5.592.3.9** `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 1255 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

**5.592.3.10** `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 1214 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::positive_sign()`.

**5.592.3.11** `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 1175 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

#### 5.592.3.12 `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 1105 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

#### 5.592.3.13 `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 1042 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_grouping()`.

---

## 5.592 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference2965

### 5.592.3.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl >::neg_format () const [inline, inherited]`

Public typedefs.

Definition at line 1145 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_neg_format()`.

### 5.592.3.15 `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 1089 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_negative_sign()`.

### 5.592.3.16 `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 1141 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_pos_format()`.

**5.592.3.17** `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 1072 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_positive_sign()`.

**5.592.3.18** `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 `moneypunct<char_type>::do_thousands_sep()`.

**Returns:**

`char_type` representing a thousands separator.

Definition at line 1012 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

## 5.592 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference 2967

### 5.592.4 Member Data Documentation

**5.592.4.1** `template<typename _CharT, bool _Intl> locale::id  
std::moneypunct<_CharT, _Intl >::id [inline, static,  
inherited]`

Numpunct facet id.

Definition at line 948 of file locale\_facets\_nonio.h.

**5.592.4.2** `template<typename _CharT, bool _Intl> const bool  
std::moneypunct_byname<_CharT, _Intl >::intl [inline,  
static]`

This value is provided by the standard, but no reason for its existence.

Reimplemented from [std::moneypunct<\\_CharT, \\_Intl >](#).

Definition at line 1324 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.593 `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

- `template<typename _Iter >`  
`move_iterator` (const `move_iterator<_Iter >` &\_\_i)
- `move_iterator` (iterator\_type \_\_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--** (int)
- `move_iterator` & **operator--** ()
- `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**



### 5.593.1 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 894 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.594 `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

- `template<typename _InputIterator >`  
`multimap` (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
`multimap` (`_InputIterator __first`, `_InputIterator __last`)
- `multimap` (`initializer_list< value_type > __l`, `const _Compare &__comp=__comp`, `const allocator_type &__a=allocator_type()`)
- `multimap` (`multimap &&__x`)
- `multimap` (`const multimap &__x`)
- `multimap` (`const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `multimap` ()
- `const_iterator` `begin` () const
- `iterator` `begin` ()
- `const_iterator` `cbegin` () const
- `const_iterator` `kend` () const

## 5.594 `std::multimap< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

- void `clear` ()
- size\_type `count` (const key\_type &\_\_x) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- bool `empty` () const
- `const_iterator` `end` () const
- `iterator` `end` ()
- `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` (iterator \_\_first, iterator \_\_last)
- size\_type `erase` (const key\_type &\_\_x)
- `iterator` `erase` (iterator \_\_position)
- `const_iterator` `find` (const key\_type &\_\_x) const
- `iterator` `find` (const key\_type &\_\_x)
- allocator\_type `get_allocator` () const
- void `insert` (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `iterator` `insert` (iterator \_\_position, const value\_type &\_\_x)
- `iterator` `insert` (const value\_type &\_\_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=` (initializer\_list< value\_type > \_\_l)
- `multimap` & `operator=` (multimap &&\_\_x)
- `multimap` & `operator=` (const multimap &\_\_x)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- size\_type `size` () const
- void `swap` (multimap &\_\_x)
- `const_iterator` `upper_bound` (const key\_type &\_\_x) const
- `iterator` `upper_bound` (const key\_type &\_\_x)
- 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 > &)`

### 5.594.1 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 86 of file `stl_multimap.h`.

### 5.594.2 Constructor & Destructor Documentation

**5.594.2.1** `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 148 of file `stl_multimap.h`.

## 5.594 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

**5.594.2.2** `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 157 of file `stl_multimap.h`.

**5.594.2.3** `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 168 of file `stl_multimap.h`.

**5.594.2.4** `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 179 of file `stl_multimap.h`.

```

5.594.2.5 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 NlogN otherwise (where N is `__l.size()`).

Definition at line 192 of file `stl_multimap.h`.

```

5.594.2.6 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 209 of file `stl_multimap.h`.

```

5.594.2.7 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.

## 5.594 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

### 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 225 of file `stl_multimap.h`.

### 5.594.3 Member Function Documentation

**5.594.3.1** `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 313 of file `stl_multimap.h`.

**5.594.3.2** `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 304 of file `stl_multimap.h`.

**5.594.3.3** `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 377 of file `stl_multimap.h`.

**5.594.3.4** `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 386 of file `stl_multimap.h`.

**5.594.3.5** `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 601 of file `stl_multimap.h`.

**5.594.3.6** `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 658 of file `stl_multimap.h`.

**5.594.3.7** `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 395 of file `stl_multimap.h`.



## 5.594 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.594.3.8** `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 404 of file `stl_multimap.h`.

**5.594.3.9** `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 411 of file `stl_multimap.h`.

**5.594.3.10** `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 331 of file `stl_multimap.h`.

**5.594.3.11** `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 322 of file `stl_multimap.h`.

**5.594.3.12** `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 742 of file stl\_multimap.h.

**5.594.3.13** `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 725 of file stl\_multimap.h.

## 5.594 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.594.3.14** `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(iterator __first, 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 558 of file stl\_multimap.h.

**5.594.3.15** `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 539 of file stl\_multimap.h.

```
5.594.3.16 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 (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 509 of file `stl_multimap.h`.

```
5.594.3.17 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 649 of file `stl_multimap.h`.

## 5.594 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.594.3.18** `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 634 of file `stl_multimap.h`.

**5.594.3.19** `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 294 of file `stl_multimap.h`.

**5.594.3.20** `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 488 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

```
5.594.3.21 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 476 of file stl\_multimap.h.

```
5.594.3.22 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 (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>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 462 of file stl\_multimap.h.

## 5.594 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

**5.594.3.23** `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 438 of file `stl_multimap.h`.

**5.594.3.24** `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 610 of file `stl_multimap.h`.

**5.594.3.25** `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 688 of file `stl_multimap.h`.

```
5.594.3.26 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 673 of file `stl_multimap.h`.

```
5.594.3.27 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 421 of file `stl_multimap.h`.

```
5.594.3.28 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](#).



## 5.594 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

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 284 of file `stl_multimap.h`.

```
5.594.3.29 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 263 of file `stl_multimap.h`.

References `std::swap()`.

```
5.594.3.30 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 248 of file `stl_multimap.h`.

**5.594.3.31** `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 349 of file `stl_multimap.h`.

**5.594.3.32** `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 340 of file `stl_multimap.h`.

**5.594.3.33** `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 367 of file `stl_multimap.h`.

**5.594.3.34** `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 358 of file `stl_multimap.h`.

## 5.594 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

**5.594.3.35** `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 416 of file `stl_multimap.h`.

**5.594.3.36** `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 591 of file `stl_multimap.h`.

Referenced by `std::swap()`.

**5.594.3.37** `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 708 of file `stl_multimap.h`.

**5.594.3.38** `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 698 of file `stl_multimap.h`.

**5.594.3.39** `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.

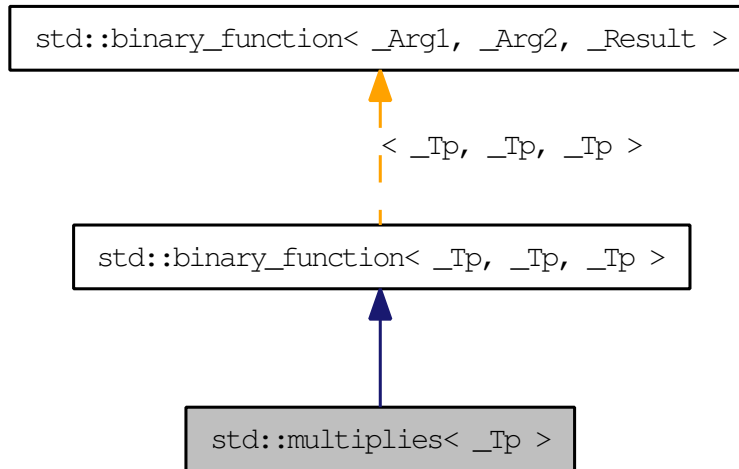
Definition at line 618 of file `stl_multimap.h`.

The documentation for this class was generated from the following file:

- [stl\\_multimap.h](#)

## 5.595 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`

#### 5.595.1 Detailed Description

```
template<typename _Tp> struct std::multiplies< _Tp >
```

One of the [math functors](#).

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

## 5.595.2 Member Typedef Documentation

**5.595.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.595.2.2** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.595.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.596 `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` (`initializer_list`< `value_type` > `__l`, `const _Compare &__comp`=`_Compare()`, `const allocator_type &__a`=`allocator_type()`)
- `multiset` (`multiset` &&`__x`)
- `multiset` (`const multiset` &`__x`)
- `template`<`typename _InputIterator` >  
`multiset` (`_InputIterator` `__first`, `_InputIterator` `__last`, `const _Compare &__comp`, `const allocator_type &__a`=`allocator_type()`)
- `template`<`typename _InputIterator` >  
`multiset` (`_InputIterator` `__first`, `_InputIterator` `__last`)
- `multiset` (`const _Compare &__comp`, `const allocator_type &__a`=`allocator_type()`)
- `multiset` ()
- `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
  - iterator `erase` (iterator \_\_first, iterator \_\_last)
  - size\_type `erase` (const key\_type &\_\_x)
  - iterator `erase` (iterator \_\_position)
  - allocator\_type `get_allocator` () const
  - void `insert` (initializer\_list< value\_type > \_\_l)
  - template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - iterator `insert` (iterator \_\_position, const value\_type &\_\_x)
  - iterator `insert` (const value\_type &\_\_x)
  - key\_compare `key_comp` () const
  - size\_type `max_size` () const
  - multiset & `operator=` (initializer\_list< value\_type > \_\_l)
  - multiset & `operator=` (multiset &&\_\_x)
  - multiset & `operator=` (const multiset &\_\_x)
  - reverse\_iterator `rbegin` () const
  - reverse\_iterator `rend` () const
  - size\_type `size` () const
  - void `swap` (multiset &\_\_x)
  - value\_compare `value_comp` () const
- 
- std::pair< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const
  - std::pair< iterator, iterator > `equal_range` (const key\_type &\_\_x)
  - 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)
  - 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 > &)



### 5.596.1 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.

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 84 of file `stl_multiset.h`.

### 5.596.2 Constructor & Destructor Documentation

```
5.596.2.1 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 129 of file `stl_multiset.h`.

```
5.596.2.2 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 138 of file `stl_multiset.h`.

**5.596.2.3** `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](#).

*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 152 of file stl\_multiset.h.

**5.596.2.4** `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 168 of file stl\_multiset.h.

**5.596.2.5** `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.

## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 2995

### 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 181 of file `stl_multiset.h`.

```
5.596.2.6 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 192 of file `stl_multiset.h`.

```
5.596.2.7 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 205 of file `stl_multiset.h`.

### 5.596.3 Member Function Documentation

**5.596.3.1** `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.

Definition at line 285 of file `stl_multiset.h`.

**5.596.3.2** `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 322 of file `stl_multiset.h`.

**5.596.3.3** `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 331 of file `stl_multiset.h`.

**5.596.3.4** `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 541 of file `stl_multiset.h`.

**5.596.3.5** `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.

## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 2997

### Parameters:

*x* Key of elements to be located.

### Returns:

Number of elements with specified key.

Definition at line 552 of file stl\_multiset.h.

**5.596.3.6** `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 340 of file stl\_multiset.h.

**5.596.3.7** `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 349 of file stl\_multiset.h.

**5.596.3.8** `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 355 of file stl\_multiset.h.

**5.596.3.9** `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 294 of file stl\_multiset.h.

**5.596.3.10** `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 636 of file stl\_multiset.h.

**5.596.3.11** `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 632 of file stl\_multiset.h.

## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 2999

**5.596.3.12** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::erase (iterator __first, 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 515 of file stl\_multiset.h.

**5.596.3.13** `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.

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 496 of file stl\_multiset.h.

**5.596.3.14** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::erase (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 466 of file `stl_multiset.h`.

**5.596.3.15** `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]`

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 574 of file `stl_multiset.h`.



## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3001

**5.596.3.16** `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.

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 570 of file `stl_multiset.h`.

**5.596.3.17** `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 276 of file `stl_multiset.h`.

**5.596.3.18** `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 445 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::insert()`.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::insert()`.

```
5.596.3.19 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 433 of file stl\_multiset.h.

```
5.596.3.20 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::insert (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 420 of file stl\_multiset.h.

**5.596.3.21** `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.

**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 396 of file `stl_multiset.h`.

**5.596.3.22** `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 268 of file `stl_multiset.h`.

**5.596.3.23** `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 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 595 of file stl\_multiset.h.

**5.596.3.24** `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 591 of file stl\_multiset.h.

**5.596.3.25** `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 365 of file stl\_multiset.h.

**5.596.3.26** `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](#).

## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3005

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 256 of file stl\_multiset.h.

```
5.596.3.27 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 235 of file stl\_multiset.h.

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

```
5.596.3.28 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 220 of file stl\_multiset.h.

```
5.596.3.29 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 303 of file `stl_multiset.h`.

**5.596.3.30** `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 312 of file `stl_multiset.h`.

**5.596.3.31** `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 360 of file `stl_multiset.h`.

**5.596.3.32** `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 380 of file `stl_multiset.h`.

Referenced by `std::swap()`.

**5.596.3.33** `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 a subsequence matching given key.

## 5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3007

### 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 611 of file stl\_multiset.h.

**5.596.3.34** `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 607 of file stl\_multiset.h.

**5.596.3.35** `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 272 of file stl\_multiset.h.

## 5.596.4 Friends And Related Function Documentation

**5.596.4.1** `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.

**5.596.4.2** `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))
```



## **5.596 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3009**

(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](#)

## 5.597 `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` ()

#### 5.597.1 Detailed Description

[mutex](#)

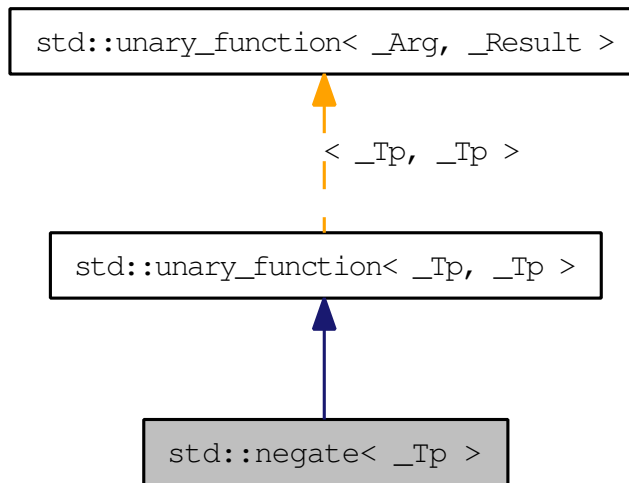
Definition at line 63 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.598 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`

### 5.598.1 Detailed Description

```
template<typename _Tp> struct std::negate< _Tp >
```

One of the [math functors](#).

Definition at line 180 of file `stl_function.h`.

## 5.598.2 Member Typedef Documentation

### 5.598.2.1 `typedef _Tp std::unary_function< _Tp , _Tp >::argument_type` [*inherited*]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.598.2.2 `typedef _Tp std::unary_function< _Tp , _Tp >::result_type` [*inherited*]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.599 `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](#) (const [param\\_type](#) &\_\_p)
- [negative\\_binomial\\_distribution](#) (`_IntType` \_\_k=1, double \_\_p=0.5)
- `_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` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & [operator](#)>> ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &, [std::negative\\_binomial\\_distribution](#)< `_IntType1` > &)

### 5.599.1 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 3340 of file random.h.

### 5.599.2 Member Typedef Documentation

**5.599.2.1** `template<typename _IntType = int> typedef _IntType  
std::negative_binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3347 of file random.h.

### 5.599.3 Member Function Documentation

**5.599.3.1** `template<typename _IntType = int> _IntType  
std::negative_binomial_distribution< _IntType >::k () const  
[inline]`

Return the  $k$  parameter of the distribution.

Definition at line 3392 of file random.h.

**5.599.3.2** `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 3428 of file random.h.

**5.599.3.3** `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 3421 of file random.h.

**5.599.3.4** `template<typename _IntType = int> double  
std::negative_binomial_distribution< _IntType >::p () const  
[inline]`

Return the  $p$  parameter of the distribution.

Definition at line 3399 of file `random.h`.

**5.599.3.5** `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 3414 of file `random.h`.

**5.599.3.6** `template<typename _IntType = int> param_type  
std::negative_binomial_distribution< _IntType >::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 3406 of file `random.h`.

**5.599.3.7** `template<typename _IntType = int> void  
std::negative_binomial_distribution< _IntType >::reset ()  
[inline]`

Resets the distribution state.

Definition at line 3385 of file `random.h`.

References `std::gamma_distribution< _RealType >::reset()`.

## 5.599.4 Friends And Related Function Documentation

**5.599.4.1** `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.

**5.599.4.2** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>>> (std::basic_istream<_CharT, _Traits > &,  
std::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](#)



## 5.600 `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

### 5.600.1 Detailed Description

```
template<typename _IntType = int> struct std::negative_binomial_
distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3349 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.601 `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__))`

### 5.601.1 Detailed Description

Exception class with `exception_ptr` data member.

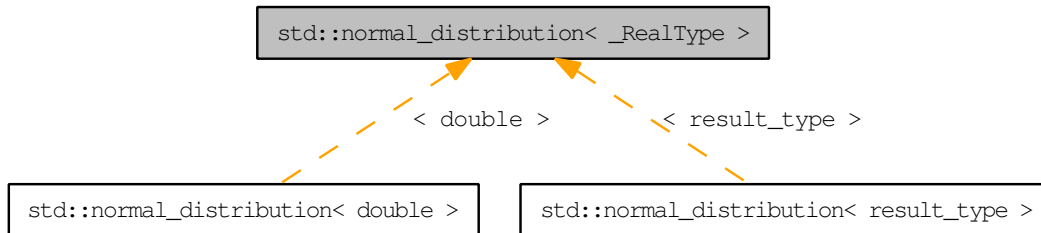
Definition at line 55 of file `nested_exception.h`.

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 5.602 `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** (const [param\\_type](#) &\_\_p)
- **normal\_distribution** ([result\\_type](#) \_\_mean=[result\\_type](#)(0), [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- [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, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>>> (std::basic_istream< _CharT, _Traits > &, std::normal_distribution< _RealType1 > &)`

### 5.602.1 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

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}$$

Definition at line 1813 of file random.h.

### 5.602.2 Member Typedef Documentation

**5.602.2.1** `template<typename _RealType = double> typedef _RealType std::normal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 1820 of file random.h.

### 5.602.3 Constructor & Destructor Documentation

**5.602.3.1** `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 1853 of file random.h.

## 5.602.4 Member Function Documentation

**5.602.4.1** `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 1910 of file `random.h`.

Referenced by `std::normal_distribution<result_type>::max()`.

**5.602.4.2** `template<typename _RealType = double> _RealType  
std::normal_distribution<_RealType>::mean() const [inline]`

Returns the mean of the distribution.

Definition at line 1874 of file `random.h`.

**5.602.4.3** `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 1903 of file `random.h`.

Referenced by `std::normal_distribution<result_type>::min()`.

**5.602.4.4** `template<typename _RealType> template<typename  
_UniformRandomNumberGenerator> normal_distribution<  
_RealType>::result_type std::normal_distribution<_RealType  
>::operator()(_UniformRandomNumberGenerator & __urng, const  
param_type & __param) [inline]`

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 1597 of file `random.tcc`.

References `std::log()`, and `std::sqrt()`.

**5.602.4.5** `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 1896 of file random.h.

**5.602.4.6** `template<typename _RealType = double> param_type  
std::normal_distribution<_RealType >::param () const [inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 1888 of file random.h.

**5.602.4.7** `template<typename _RealType = double> void  
std::normal_distribution<_RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 1867 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()`.

**5.602.4.8** `template<typename _RealType = double> _RealType  
std::normal_distribution<_RealType >::stddev () const [inline]`

Returns the standard deviation of the distribution.

Definition at line 1881 of file random.h.

**5.602.5 Friends And Related Function Documentation**

**5.602.5.1** `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.

**Returns:**

The output stream with the state of `__x` inserted or in an error state.

**5.602.5.2** `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](#)

## 5.603 `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

### 5.603.1 Detailed Description

`template<typename _RealType = double> struct std::normal_distribution<_RealType >::param_type`

Parameter type.

Definition at line 1822 of file random.h.

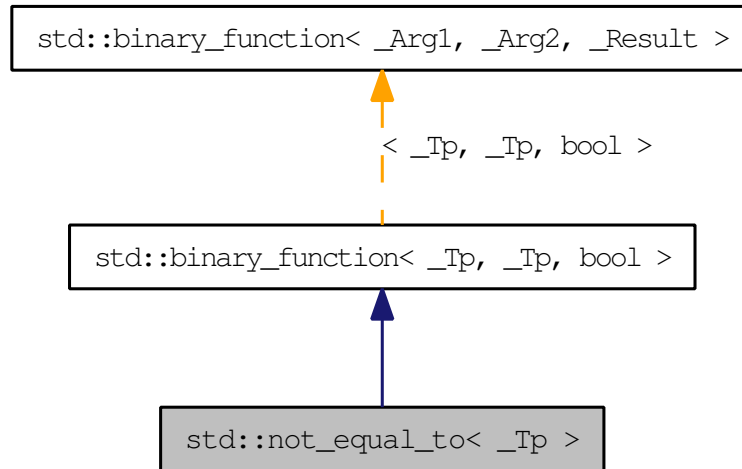
The documentation for this struct was generated from the following file:

- [random.h](#)



## 5.604 `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`

#### 5.604.1 Detailed Description

```
template<typename _Tp> struct std::not_equal_to<_Tp>
```

One of the [comparison functors](#).

Definition at line 208 of file `stl_function.h`.

## 5.604.2 Member Typedef Documentation

**5.604.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [*inherited*]

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.604.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [*inherited*]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.604.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [*inherited*]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

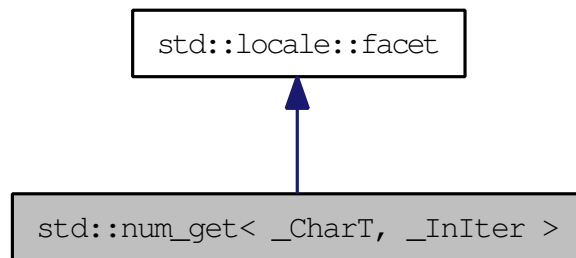
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.605 `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 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`

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~num_get` ()
- `__attribute__((__const__))` static `const char *_S_get_c_name` () `throw ()`
- `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 *_zero, size_t __len, _CharT2 __c`) `const`

- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, int >::__type _M_`  
`find (const _CharT2 *, size_t __len, _CharT2 __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 __c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::Impl`
- `typedef _CharT char_type`
- `typedef _InIter iter_type`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__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, float &__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, long long &__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, unsigned int &__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, long &__v) const`
- `virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, void *&) 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, double &) 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 __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long 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 &__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 short &__v) 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, iter_type, ios_base &, ios_base::iostate &, bool &) const`

### 5.605.1 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 1909 of file `locale_facets.h`.

### 5.605.2 Member Typedef Documentation

**5.605.2.1** `template<typename _CharT, typename _InIter > typedef _CharT std::num_get< _CharT, _InIter >::char_type`

Public typedefs.

Definition at line 1915 of file `locale_facets.h`.

**5.605.2.2** `template<typename _CharT, typename _InIter > typedef _InIter  
std::num_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1916 of file locale\_facets.h.

### 5.605.3 Constructor & Destructor Documentation

**5.605.3.1** `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 1930 of file locale\_facets.h.

**5.605.3.2** `template<typename _CharT, typename _InIter > virtual  
std::num_get< _CharT, _InIter >::~~num_get () [inline,  
protected, virtual]`

Destructor.

Definition at line 2099 of file locale\_facets.h.

### 5.605.4 Member Function Documentation

**5.605.4.1** `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, void *& __v)  
const [inline, protected, virtual]`

Public typedefs.

Definition at line 748 of file locale\_facets.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.

**5.605.4.2** `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, long double &  
__v) const [inline, protected, virtual]`

Public typedefs.

Definition at line 733 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().

**5.605.4.3** `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 [inline, protected, virtual]`

Public typedefs.

Definition at line 701 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().

**5.605.4.4** `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  
[inline, protected, virtual]`

Public typedefs.

Definition at line 686 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().

**5.605.4.5** `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 long  
long & __v) const [inline, protected, virtual]`

Public typedefs.

Definition at line 2193 of file locale\_facets.h.

**5.605.4.6** `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, long long & __v)  
const [inline, protected, virtual]`

Public typedefs.

Definition at line 2188 of file locale\_facets.h.

**5.605.4.7** `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, unsigned long &  
__v) const [inline, protected, virtual]`

Public typedefs.

Definition at line 2182 of file locale\_facets.h.

**5.605.4.8** `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 int &  
__v) const [inline, protected, virtual]`

Public typedefs.

Definition at line 2177 of file locale\_facets.h.

**5.605.4.9** `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]`

Public typedefs.

Definition at line 2172 of file locale\_facets.h.

**5.605.4.10** `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]`

Public typedefs.

Definition at line 2167 of file locale\_facets.h.



**5.605.4.11** `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, bool &  
__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 590 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\(\)](#).

**5.605.4.12** `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, 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 [numpunct::grouping\(\)](#) and [numpunct::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 2093 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

**5.605.4.13** `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, long double &  
__v) const [inline]`

Public typedefs.

Definition at line 2061 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

**5.605.4.14** `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, double & __v)  
const [inline]`

Public typedefs.

Definition at line 2056 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

**5.605.4.15** `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, 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 `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 2051 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.605.4.16 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, unsigned long
long & __v) const [inline]
```

Public typedefs.

Definition at line 2018 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.605.4.17 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, long long & __v)
const [inline]
```

Public typedefs.

Definition at line 2013 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.605.4.18** `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 &  
__v) const [inline]`

Public typedefs.

Definition at line 2007 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.605.4.19** `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]`

Public typedefs.

Definition at line 2002 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.605.4.20** `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, unsigned short &  
__v) const [inline]`

Public typedefs.

Definition at line 1997 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.605.4.21** `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 & __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 [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 1992 of file locale\_facets.h.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

```
5.605.4.22 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, 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 `cctype<CharT>::truenamename()` or `cctype<CharT>::falsenamename()`. 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 1956 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

Referenced by `std::basic_istream<_CharT, _Traits >::operator>>()`.

### 5.605.5 Member Data Documentation

#### 5.605.5.1 `template<typename _CharT, typename _InIter > locale::id` `std::num_get<_CharT, _InIter >::id [inline, static]`

Numpunct facet id.

Definition at line 1920 of file locale\_facets.h.

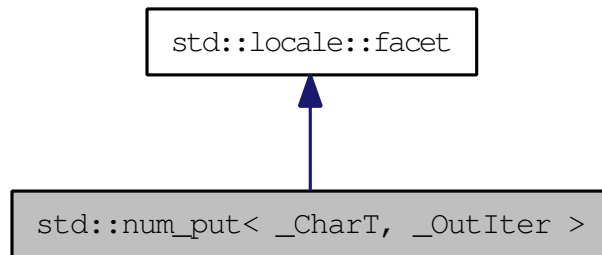
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

## 5.606 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 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

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~num\\_put](#) ()
- [\\_\\_attribute\\_\\_](#) ((\_\_const\_\_) static const char \*\_S\_get\_c\_name() throw ())
- 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`

## 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 __c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- `class locale::Impl`
- `typedef _CharT char_type`
- `typedef _OutIter iter_type`
- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, long double __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, unsigned long 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 __v) const`
- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, long __v) const`
- `virtual iter_type do_put (iter_type, ios_base &, char_type __fill, const void *__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, double __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 __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 __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, ios_base &, char_type __fill, bool __v) const`

### 5.606.1 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 2247 of file `locale_facets.h`.

### 5.606.2 Member Typedef Documentation

**5.606.2.1 template<typename \_CharT , typename \_OutIter > typedef \_CharT std::num\_put< \_CharT, \_OutIter >::char\_type**

Public typedefs.

Definition at line 2253 of file `locale_facets.h`.

**5.606.2.2 template<typename \_CharT , typename \_OutIter > typedef \_OutIter std::num\_put< \_CharT, \_OutIter >::iter\_type**

Public typedefs.

Definition at line 2254 of file `locale_facets.h`.

### 5.606.3 Constructor & Destructor Documentation

**5.606.3.1 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 2268 of file locale\_facets.h.

**5.606.3.2** `template<typename _CharT, typename _OutIter > virtual  
std::num_put< _CharT, _OutIter >::~~num_put () [inline,  
protected, virtual]`

Destructor.

Definition at line 2447 of file locale\_facets.h.

**5.606.4 Member Function Documentation**

**5.606.4.1** `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 [inline,  
protected, virtual]`

Public typedefs.

Definition at line 1162 of file locale\_facets.tcc.

References `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::uppercase`.

**5.606.4.2** `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 [inline,  
protected, virtual]`

Public typedefs.

Definition at line 1155 of file locale\_facets.tcc.

**5.606.4.3** `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 [inline,  
protected, virtual]`

Public typedefs.

Definition at line 1141 of file locale\_facets.tcc.

**5.606.4.4** `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]`

Public typedefs.

Definition at line 2482 of file locale\_facets.h.

**5.606.4.5** `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]`

Public typedefs.

Definition at line 2477 of file locale\_facets.h.

**5.606.4.6** `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]`

Public typedefs.

Definition at line 2471 of file locale\_facets.h.

**5.606.4.7** `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]`

Public typedefs.

Definition at line 2467 of file locale\_facets.h.

**5.606.4.8** `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 [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 1089 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()`.

**5.606.4.9** `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 2416 of file `locale_facets.h`.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**5.606.4.10** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put<_CharT, _OutIter >::put (iter_type __s, ios_base &  
__f, char_type __fill, long double __v) const [inline]`

Public typedefs.

Definition at line 2395 of file locale\_facets.h.

References std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.606.4.11** `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 2391 of file locale\_facets.h.

References std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.606.4.12** `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]`

Public typedefs.

Definition at line 2342 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**5.606.4.13** `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]`

Public typedefs.

Definition at line 2338 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**5.606.4.14** `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]`

Public typedefs.

Definition at line 2332 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**5.606.4.15** `template<typename _CharT, typename _OutIter> iter_type  
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base &  
__f, char_type __fill, long __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 2328 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**5.606.4.16** `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 2286 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.606.5 Member Data Documentation

### 5.606.5.1 `template<typename _CharT, typename _OutIter > locale::id` `std::num_put<_CharT, _OutIter >::id` [`inline`, `static`]

Numpunct facet id.

Definition at line 2258 of file `locale_facets.h`.

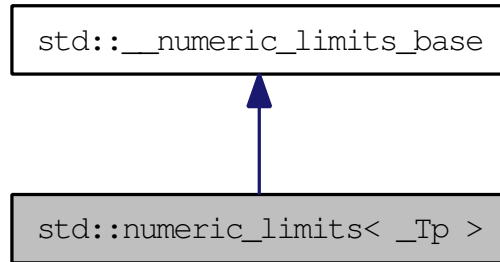
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)



## 5.607 std::numeric\_limits< \_Tp > Struct Template Reference

Properties of fundamental types. Inheritance diagram for std::numeric\_limits< \_Tp >:



### Static Public Member Functions

- static `_Tp` [denorm\\_min](#) () throw ()
- static `_Tp` [epsilon](#) () throw ()
- static `_Tp` [infinity](#) () throw ()
- static `_Tp` [lowest](#) () throw ()
- static `_Tp` [max](#) () throw ()
- static `_Tp` [min](#) () throw ()
- static `_Tp` [quiet\\_NaN](#) () throw ()
- static `_Tp` [round\\_error](#) () throw ()
- static `_Tp` [signaling\\_NaN](#) () throw ()

### Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const `float_denorm_style` [has\\_denorm](#)
- static const bool [has\\_denorm\\_loss](#)
- static const bool [has\\_infinity](#)
- static const bool [has\\_quiet\\_NaN](#)
- static const bool [has\\_signaling\\_NaN](#)
- static const bool [is\\_bounded](#)
- static const bool [is\\_exact](#)
- static const bool [is\\_iec559](#)
- static const bool [is\\_integer](#)
- static const bool [is\\_modulo](#)

- static const bool [is\\_signed](#)
- static const bool [is\\_specialized](#)
- static const int [max\\_digits10](#)
- static const int [max\\_exponent](#)
- static const int [max\\_exponent10](#)
- static const int [min\\_exponent](#)
- static const int [min\\_exponent10](#)
- static const int [radix](#)
- static const [float\\_round\\_style](#) [round\\_style](#)
- static const bool [tinyness\\_before](#)
- static const bool [traps](#)

### 5.607.1 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 284 of file limits.

### 5.607.2 Member Function Documentation

**5.607.2.1 template<typename \_Tp > static \_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 313 of file limits.

**5.607.2.2 template<typename \_Tp > static \_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 298 of file limits.

**5.607.2.3** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::infinity () throw () [inline, static]`

The representation of positive infinity, if `has_infinity`.

Definition at line 302 of file limits.

**5.607.2.4** `template<typename _Tp > static _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 294 of file limits.

**5.607.2.5** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::max () throw () [inline, static]`

The maximum finite value.

Definition at line 290 of file limits.

**5.607.2.6** `template<typename _Tp > static _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 288 of file limits.

**5.607.2.7** `template<typename _Tp > static _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 306 of file limits.

**5.607.2.8** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::round_error () throw () [inline, static]`

The maximum rounding error measurement (see LIA-1).

Definition at line 300 of file limits.

**5.607.2.9** `template<typename _Tp > static _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 309 of file `limits`.

### 5.607.3 Member Data Documentation

**5.607.3.1** `const 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 199 of file `limits`.

**5.607.3.2** `const int std::__numeric_limits_base::digits10 [static, inherited]`

The number of base 10 digits that can be represented without change.

Definition at line 201 of file `limits`.

**5.607.3.3** `const float_denorm_style std::__numeric_limits_base::has_denorm [static, inherited]`

See [std::float\\_denorm\\_style](#) for more information.

Definition at line 244 of file `limits`.

**5.607.3.4** `const 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 247 of file `limits`.

**5.607.3.5** `const bool std::__numeric_limits_base::has_infinity [static, inherited]`

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

**5.607.3.6** `const 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 239 of file limits.

**5.607.3.7** `const 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 242 of file limits.

**5.607.3.8** `const 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 255 of file limits.

**5.607.3.9** `const 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 216 of file limits.

**5.607.3.10** `const 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 251 of file limits.

**5.607.3.11** `const 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 211 of file limits.

**5.607.3.12** `const 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 260 of file limits.

**5.607.3.13** `const bool std::__numeric_limits_base::is_signed` [`static`, `inherited`]

True if the type is signed.

Definition at line 208 of file limits.

**5.607.3.14** `const 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 194 of file limits.

**5.607.3.15** `const 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 205 of file limits.

**5.607.3.16** `const 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 230 of file `limits`.

**5.607.3.17** `const 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 233 of file `limits`.

**5.607.3.18** `const 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 223 of file `limits`.

**5.607.3.19** `const 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 226 of file `limits`.

**5.607.3.20** `const 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 219 of file `limits`.

**5.607.3.21** `const 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 269 of file `limits`.

**5.607.3.22** `const bool std::__numeric_limits_base::tinyness_before` [`static`, `inherited`]

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file limits.

**5.607.3.23** `const bool std::__numeric_limits_base::traps` [`static`, `inherited`]

True if trapping is implemented for this type.

Definition at line 263 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.608 `std::numeric_limits< bool >` Struct Template Reference

[numeric\\_limits<bool>](#) specialization.

### Static Public Member Functions

- static bool **denorm\_min** () throw ()
- static bool **epsilon** () throw ()
- static bool **infinity** () throw ()
- static bool **lowest** () throw ()
- static bool **max** () throw ()
- static bool **min** () throw ()
- static bool **quiet\_NaN** () throw ()
- static bool **round\_error** () throw ()
- static bool **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.608.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

[numeric\\_limits<bool>](#) specialization.

Definition at line 335 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.609 `std::numeric_limits< char >` Struct Template Reference

[numeric\\_limits<char>](#) specialization.

### Static Public Member Functions

- static char **denorm\_min** () throw ()
- static char **epsilon** () throw ()
- static char **infinity** () throw ()
- static char **lowest** () throw ()
- static char **max** () throw ()
- static char **min** () throw ()
- static char **quiet\_NaN** () throw ()
- static char **round\_error** () throw ()
- static char **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.609.1 Detailed Description

`template<> struct std::numeric_limits< char >`

[numeric\\_limits<char>](#) specialization.

Definition at line 395 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.610 `std::numeric_limits< char16_t >` Struct Template Reference

[numeric\\_limits<char16\\_t>](#) specialization.

### Static Public Member Functions

- static `char16_t` **denorm\_min** () throw ()
- static `char16_t` **epsilon** () throw ()
- static `char16_t` **infinity** () throw ()
- static `char16_t` **lowest** () throw ()
- static `char16_t` **max** () throw ()
- static `char16_t` **min** () throw ()
- static `char16_t` **quiet\_NaN** () throw ()
- static `char16_t` **round\_error** () throw ()
- static `char16_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.610.1 Detailed Description

`template<> struct std::numeric_limits< char16_t >`

[numeric\\_limits<char16\\_t>](#) specialization.

Definition at line 628 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.611 `std::numeric_limits< char32_t >` Struct Template Reference

[numeric\\_limits<char32\\_t>](#) specialization.

### Static Public Member Functions

- static `char32_t` **denorm\_min** () throw ()
- static `char32_t` **epsilon** () throw ()
- static `char32_t` **infinity** () throw ()
- static `char32_t` **lowest** () throw ()
- static `char32_t` **max** () throw ()
- static `char32_t` **min** () throw ()
- static `char32_t` **quiet\_NaN** () throw ()
- static `char32_t` **round\_error** () throw ()
- static `char32_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const `float_denorm_style` **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const `float_round_style` **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.611.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

[numeric\\_limits<char32\\_t>](#) specialization.

Definition at line 686 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.612 `std::numeric_limits< double >` Struct Template Reference

[numeric\\_limits<double>](#) specialization.

### Static Public Member Functions

- static double **denorm\_min** () throw ()
- static double **epsilon** () throw ()
- static double **infinity** () throw ()
- static double **lowest** () throw ()
- static double **max** () throw ()
- static double **min** () throw ()
- static double **quiet\_NaN** () throw ()
- static double **round\_error** () throw ()
- static double **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.612.1 Detailed Description

`template<> struct std::numeric_limits< double >`

[numeric\\_limits<double>](#) specialization.

Definition at line 1274 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.613 `std::numeric_limits< float >` Struct Template Reference

[numeric\\_limits<float>](#) specialization.

### Static Public Member Functions

- static float **denorm\_min** () throw ()
- static float **epsilon** () throw ()
- static float **infinity** () throw ()
- static float **lowest** () throw ()
- static float **max** () throw ()
- static float **min** () throw ()
- static float **quiet\_NaN** () throw ()
- static float **round\_error** () throw ()
- static float **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.613.1 Detailed Description

`template<> struct std::numeric_limits< float >`

[numeric\\_limits<float>](#) specialization.

Definition at line 1209 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.614 `std::numeric_limits<int>` Struct Template Reference

[numeric\\_limits<int>](#) specialization.

### Static Public Member Functions

- static int **denorm\_min** () throw ()
- static int **epsilon** () throw ()
- static int **infinity** () throw ()
- static int **lowest** () throw ()
- static int **max** () throw ()
- static int **min** () throw ()
- static int **quiet\_NaN** () throw ()
- static int **round\_error** () throw ()
- static int **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.614.1 Detailed Description

`template<> struct std::numeric_limits< int >`

[numeric\\_limits<int>](#) specialization.

Definition at line 861 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.615 `std::numeric_limits< long >` Struct Template Reference

[numeric\\_limits<long>](#) specialization.

### Static Public Member Functions

- static long **denorm\_min** () throw ()
- static long **epsilon** () throw ()
- static long **infinity** () throw ()
- static long **lowest** () throw ()
- static long **max** () throw ()
- static long **min** () throw ()
- static long **quiet\_NaN** () throw ()
- static long **round\_error** () throw ()
- static long **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.615.1 Detailed Description

`template<> struct std::numeric_limits< long >`

[numeric\\_limits<long>](#) specialization.

Definition at line 977 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.616 `std::numeric_limits< long double >` Struct Template Reference

[numeric\\_limits<long double>](#) specialization.

### Static Public Member Functions

- static long double **denorm\_min** () throw ()
- static long double **epsilon** () throw ()
- static long double **infinity** () throw ()
- static long double **lowest** () throw ()
- static long double **max** () throw ()
- static long double **min** () throw ()
- static long double **quiet\_NaN** () throw ()
- static long double **round\_error** () throw ()
- static long double **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.616.1 Detailed Description

`template<> struct std::numeric_limits< long double >`

[numeric\\_limits<long double>](#) specialization.

Definition at line 1339 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.617 `std::numeric_limits< long long >` Struct Template Reference

[numeric\\_limits<long long>](#) specialization.

### Static Public Member Functions

- static long long **denorm\_min** () throw ()
- static long long **epsilon** () throw ()
- static long long **infinity** () throw ()
- static long long **lowest** () throw ()
- static long long **max** () throw ()
- static long long **min** () throw ()
- static long long **quiet\_NaN** () throw ()
- static long long **round\_error** () throw ()
- static long long **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.617.1 Detailed Description

`template<> struct std::numeric_limits< long long >`

[numeric\\_limits<long long>](#) specialization.

Definition at line 1093 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.618 `std::numeric_limits< short >` Struct Template Reference

[numeric\\_limits<short>](#) specialization.

### Static Public Member Functions

- static short **denorm\_min** () throw ()
- static short **epsilon** () throw ()
- static short **infinity** () throw ()
- static short **lowest** () throw ()
- static short **max** () throw ()
- static short **min** () throw ()
- static short **quiet\_NaN** () throw ()
- static short **round\_error** () throw ()
- static short **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.618.1 Detailed Description

`template<> struct std::numeric_limits< short >`

[numeric\\_limits<short>](#) specialization.

Definition at line 745 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.619 `std::numeric_limits< signed char >` Struct Template Reference

[numeric\\_limits<signed char>](#) specialization.

### Static Public Member Functions

- static signed char **denorm\_min** () throw ()
- static signed char **epsilon** () throw ()
- static signed char **infinity** () throw ()
- static signed char **lowest** () throw ()
- static signed char **max** () throw ()
- static signed char **min** () throw ()
- static signed char **quiet\_NaN** () throw ()
- static signed char **round\_error** () throw ()
- static signed char **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.619.1 Detailed Description

`template<> struct std::numeric_limits< signed char >`

[numeric\\_limits<signed char>](#) specialization.

Definition at line 453 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.620 `std::numeric_limits< unsigned char >` Struct Template Reference

[numeric\\_limits<unsigned char>](#) specialization.

### Static Public Member Functions

- static unsigned char **denorm\_min** () throw ()
- static unsigned char **epsilon** () throw ()
- static unsigned char **infinity** () throw ()
- static unsigned char **lowest** () throw ()
- static unsigned char **max** () throw ()
- static unsigned char **min** () throw ()
- static unsigned char **quiet\_NaN** () throw ()
- static unsigned char **round\_error** () throw ()
- static unsigned char **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.620.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

[numeric\\_limits<unsigned char>](#) specialization.

Definition at line 511 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.621 `std::numeric_limits< unsigned int >` Struct Template Reference

[numeric\\_limits<unsigned int>](#) specialization.

### Static Public Member Functions

- static unsigned int **denorm\_min** () throw ()
- static unsigned int **epsilon** () throw ()
- static unsigned int **infinity** () throw ()
- static unsigned int **lowest** () throw ()
- static unsigned int **max** () throw ()
- static unsigned int **min** () throw ()
- static unsigned int **quiet\_NaN** () throw ()
- static unsigned int **round\_error** () throw ()
- static unsigned int **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.621.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

[numeric\\_limits<unsigned int>](#) specialization.

Definition at line 919 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.622 `std::numeric_limits< unsigned long >` Struct Template Reference

[numeric\\_limits<unsigned long>](#) specialization.

### Static Public Member Functions

- static unsigned long **denorm\_min** () throw ()
- static unsigned long **epsilon** () throw ()
- static unsigned long **infinity** () throw ()
- static unsigned long **lowest** () throw ()
- static unsigned long **max** () throw ()
- static unsigned long **min** () throw ()
- static unsigned long **quiet\_NaN** () throw ()
- static unsigned long **round\_error** () throw ()
- static unsigned long **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.622.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long >`

[numeric\\_limits<unsigned long>](#) specialization.

Definition at line 1035 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.623 `std::numeric_limits< unsigned long long >` Struct Template Reference

[numeric\\_limits<unsigned long long>](#) specialization.

### Static Public Member Functions

- static unsigned long long **denorm\_min** () throw ()
- static unsigned long long **epsilon** () throw ()
- static unsigned long long **infinity** () throw ()
- static unsigned long long **lowest** () throw ()
- static unsigned long long **max** () throw ()
- static unsigned long long **min** () throw ()
- static unsigned long long **quiet\_NaN** () throw ()
- static unsigned long long **round\_error** () throw ()
- static unsigned long long **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.623.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

[numeric\\_limits<unsigned long long>](#) specialization.

Definition at line 1151 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.624 `std::numeric_limits< unsigned short >` Struct Template Reference

[numeric\\_limits<unsigned short>](#) specialization.

### Static Public Member Functions

- static unsigned short **denorm\_min** () throw ()
- static unsigned short **epsilon** () throw ()
- static unsigned short **infinity** () throw ()
- static unsigned short **lowest** () throw ()
- static unsigned short **max** () throw ()
- static unsigned short **min** () throw ()
- static unsigned short **quiet\_NaN** () throw ()
- static unsigned short **round\_error** () throw ()
- static unsigned short **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.624.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

[numeric\\_limits<unsigned short>](#) specialization.

Definition at line 803 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.625 `std::numeric_limits< wchar_t >` Struct Template Reference

[numeric\\_limits<wchar\\_t>](#) specialization.

### Static Public Member Functions

- static `wchar_t` **denorm\_min** () throw ()
- static `wchar_t` **epsilon** () throw ()
- static `wchar_t` **infinity** () throw ()
- static `wchar_t` **lowest** () throw ()
- static `wchar_t` **max** () throw ()
- static `wchar_t` **min** () throw ()
- static `wchar_t` **quiet\_NaN** () throw ()
- static `wchar_t` **round\_error** () throw ()
- static `wchar_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.625.1 Detailed Description

`template<> struct std::numeric_limits< wchar_t >`

[numeric\\_limits<wchar\\_t>](#) specialization.

Definition at line 569 of file limits.

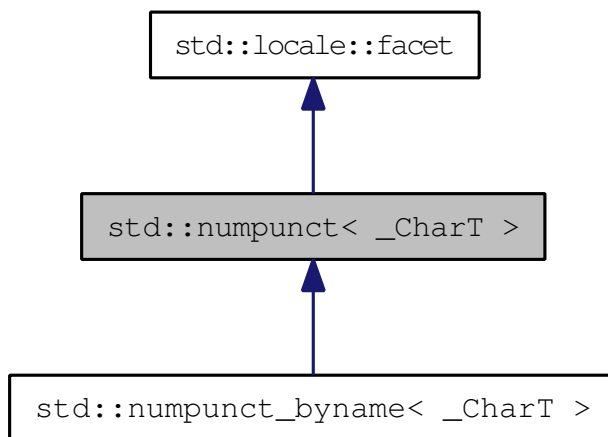
The documentation for this struct was generated from the following file:

- [limits](#)

## 5.626 `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` (`__c_locale __cloc`, `size_t __refs=0`)
- `numpunct` (`__cache_type *__cache`, `size_t __refs=0`)
- `numpunct` (`size_t __refs=0`)
- `template<>`  
void `_M_initialize_numpunct` (`__c_locale __cloc`)
- `template<>`  
void `_M_initialize_numpunct` (`__c_locale __cloc`)
- `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](#) ()
- [\\_\\_attribute\\_\\_](#) ((\_\_const\_\_)) static const char \*\_S\_get\_c\_name() throw ()
- void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc=NULL)
- 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 \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- [\\_\\_cache\\_type \\* \\_M\\_data](#)

### Friends

- class [locale::\\_Impl](#)

### 5.626.1 Detailed Description

**template<typename \_CharT> class std::num\_punct<\_CharT>**

Primary class template [num\\_punct](#).

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 [num\\_punct](#) facet is used by streams for many I/O operations involving numbers. The [num\\_punct](#) 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 [num\\_punct](#) facet.

Definition at line 1636 of file locale\_facets.h.

### 5.626.2 Member Typedef Documentation

**5.626.2.1 template<typename \_CharT> typedef \_CharT std::num\_punct<\_CharT>::char\_type**

Public typedefs.

Reimplemented in [std::num\\_punct\\_byname<\\_CharT>](#).

Definition at line 1642 of file locale\_facets.h.

**5.626.2.2 template<typename \_CharT> typedef basic\_string<\_CharT> std::num\_punct<\_CharT>::string\_type**

Public typedefs.

Reimplemented in [std::num\\_punct\\_byname<\\_CharT>](#).

Definition at line 1643 of file locale\_facets.h.

### 5.626.3 Constructor & Destructor Documentation

**5.626.3.1 template<typename \_CharT> std::num\_punct<\_CharT>::num\_punct(size\_t \_\_refs = 0) [inline, explicit]**

Num\_punct constructor.

#### Parameters:

*refs* Refcount to pass to the base class.

Definition at line 1660 of file locale\_facets.h.

```
5.626.3.2 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 1673 of file locale\_facets.h.

```
5.626.3.3 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 1687 of file locale\_facets.h.

```
5.626.3.4 template<typename _CharT > virtual std::num_punct< _CharT
>::~~num_punct () [protected, virtual]
```

Destructor.

## 5.626.4 Member Function Documentation

```
5.626.4.1 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 `num_punct<char_type>::do_decimal_point()`.



**Returns:**

*char\_type* representing a [decimal](#) point.

Definition at line 1701 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_decimal\_point().

**5.626.4.2** `template<typename _CharT> virtual char_type std::num_punct<_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 1788 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::decimal\_point().

**5.626.4.3** `template<typename _CharT> virtual string_type std::num_punct<_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 1839 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::falsename().

**5.626.4.4** `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 1813 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::grouping()`.

**5.626.4.5** `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 1800 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.

**5.626.4.6** `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 1826 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::truename()`.

**5.626.4.7** `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 1771 of file locale\_facets.h.

References [std::num\\_punct<\\_CharT>::do\\_falsename\(\)](#).

**5.626.4.8 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 1745 of file locale\_facets.h.

References [std::num\\_punct<\\_CharT>::do\\_grouping\(\)](#).

**5.626.4.9 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 1714 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_thousands_sep()`.

#### **5.626.4.10** `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 1758 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_true_name()`.

### **5.626.5 Member Data Documentation**

#### **5.626.5.1** `template<typename _CharT> locale::id std::num_punct<_CharT>::id [inline, static]`

Numpunct facet id.

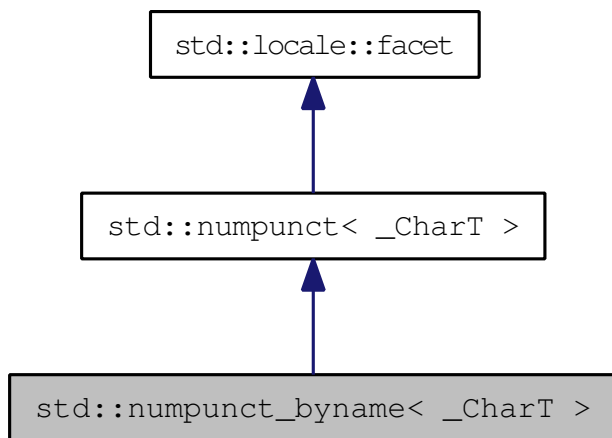
Definition at line 1652 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.627 `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`)
- `template<>`  
`void _M_initialize_numpunct (__c_locale __cloc)`
- `template<>`  
`void _M_initialize_numpunct (__c_locale __cloc)`
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string_type grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- `__attribute__((const))` static const char \*\_S\_get\_c\_name() throw ()
- void `_M_initialize_numpunct` (\_\_c\_locale \_\_cloc=NULL)
- 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 \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__cache_type * _M_data`

## Friends

- class `locale::_Impl`

### 5.627.1 Detailed Description

`template<typename _CharT> class std::numpunct_byname<_CharT >`

class [numpunct\\_byname](#) [22.2.3.2].

Definition at line 1868 of file `locale_facets.h`.

## 5.627.2 Member Typedef Documentation

### 5.627.2.1 `template<typename _CharT> typedef _CharT std::num_punct_byname<_CharT>::char_type`

Public typedefs.

Reimplemented from [std::num\\_punct<\\_CharT>](#).

Definition at line 1871 of file `locale_facets.h`.

### 5.627.2.2 `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 1872 of file `locale_facets.h`.

## 5.627.3 Member Function Documentation

### 5.627.3.1 `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 `num_punct<char_type>::do_decimal_point()`.

#### Returns:

*char\_type* representing a [decimal](#) point.

Definition at line 1701 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_decimal_point()`.

### 5.627.3.2 `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 1788 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

**5.627.3.3** `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 1839 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::falsename()`.

**5.627.3.4** `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 1813 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::grouping()`.

**5.627.3.5** `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 1800 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.

**5.627.3.6 `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 1826 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::truename()`.

**5.627.3.7 `template<typename _CharT> string_type std::num_punct<_CharT>::do_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 1771 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_falsename()`.

**5.627.3.8 `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 1745 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_grouping()`.

**5.627.3.9 `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 1714 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_thousands_sep()`.

**5.627.3.10 `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 1758 of file locale\_facets.h.

References `std::numpunct<_CharT>::do_truename()`.

#### **5.627.4 Member Data Documentation**

##### **5.627.4.1 `template<typename _CharT> locale::id std::numpunct<_CharT>::id` [`inline`, `static`, `inherited`]**

Numpunct facet id.

Definition at line 1652 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.628 `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)

### 5.628.1 Detailed Description

[once\\_flag](#)

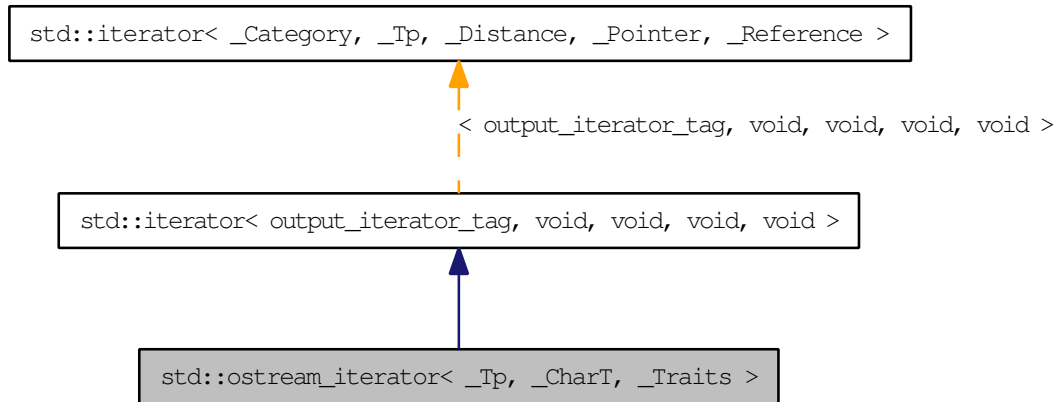
Definition at line 676 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.629 `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 `basic_ostream< _CharT, _Traits >` [ostream\\_type](#)
  - typedef `_Traits` [traits\\_type](#)

### Public Member Functions

- [ostream\\_iterator](#) (`const ostream_iterator &__obj`)
- [ostream\\_iterator](#) (`ostream_type &__s, const _CharT *__c`)
- [ostream\\_iterator](#) (`ostream_type &__s`)
- [ostream\\_iterator](#) & **operator\*** ()
- [ostream\\_iterator](#) & **operator++** (int)
- [ostream\\_iterator](#) & **operator++** ()
- [ostream\\_iterator](#) & **operator=** (`const _Tp &__value`)

### 5.629.1 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 152 of file `stream_iterator.h`.

### 5.629.2 Member Typedef Documentation

**5.629.2.1** `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 158 of file `stream_iterator.h`.

**5.629.2.2** `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 115 of file `stl_iterator_base_types.h`.

**5.629.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 111 of file `stl_iterator_base_types.h`.

## 5.629 std::ostream\_iterator< \_Tp, \_CharT, \_Traits > Class Template Reference

**5.629.2.4** `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 160 of file stream\_iterator.h.

**5.629.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 117 of file stl\_iterator\_base\_types.h.

**5.629.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file stl\_iterator\_base\_types.h.

**5.629.2.7** `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 159 of file stream\_iterator.h.

**5.629.2.8** `typedef void std::iterator< output_iterator_tag, void, void, void,  
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file stl\_iterator\_base\_types.h.

### 5.629.3 Constructor & Destructor Documentation

**5.629.3.1** `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 169 of file stream\_iterator.h.

**5.629.3.2** `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 181 of file stream\_iterator.h.

**5.629.3.3** `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 185 of file stream\_iterator.h.

### 5.629.4 Member Function Documentation

**5.629.4.1** `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.



### **5.629 std::ostream\_iterator< \_Tp, \_CharT, \_Traits > Class Template Reference**

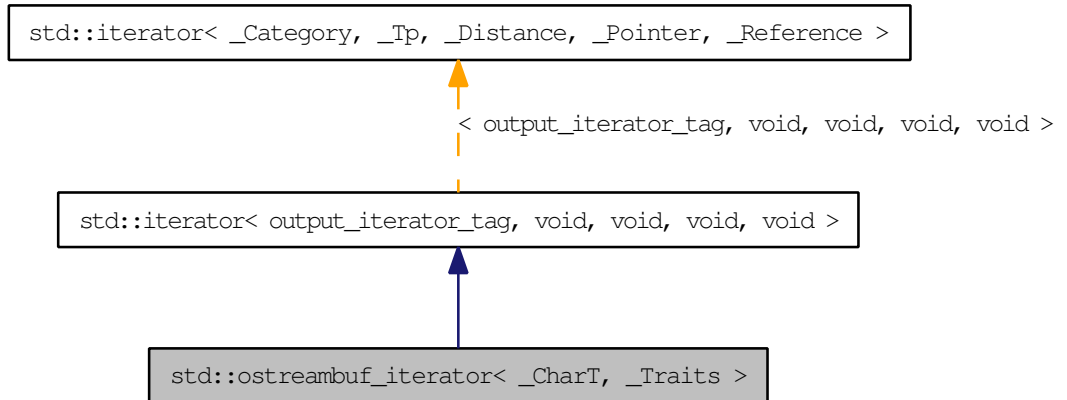
Definition at line 191 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.630 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

Provides output [iterator](#) semantics for streambufs. 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 [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)
- typedef [basic\\_streambuf< \\_CharT, \\_Traits >](#) [streambuf\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)

### Public Member Functions

- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) throw ()
- [ostreambuf\\_iterator](#) ([ostream\\_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)

## 5.630 `std::ostreambuf_iterator<_CharT, _Traits>` Class Template Reference 3115

### 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 >)`

### 5.630.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::ostreambuf_iterator<_CharT, _Traits>`

Provides output [iterator](#) semantics for streambufs.

Definition at line 204 of file `streambuf_iterator.h`.

### 5.630.2 Member Typedef Documentation

**5.630.2.1** `template<typename _CharT, typename _Traits> typedef _CharT std::ostreambuf_iterator<_CharT, _Traits>::char_type`

Public typedefs.

Definition at line 211 of file `streambuf_iterator.h`.

**5.630.2.2** `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 115 of file `stl_iterator_base_types.h`.

**5.630.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 111 of file `stl_iterator_base_types.h`.

**5.630.2.4** `template<typename _CharT, typename _Traits> typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator<_CharT, _Traits>::ostream_type`

Public typedefs.

Definition at line 214 of file `streambuf_iterator.h`.

**5.630.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.630.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file `stl_iterator_base_types.h`.

**5.630.2.7** `template<typename _CharT , typename _Traits > typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator<_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 213 of file `streambuf_iterator.h`.

**5.630.2.8** `template<typename _CharT , typename _Traits > typedef _Traits std::ostreambuf_iterator<_CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 212 of file `streambuf_iterator.h`.

**5.630.2.9** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file `stl_iterator_base_types.h`.

## 5.630 std::ostreambuf\_iterator<\_CharT, \_Traits> Class Template Reference

### 5.630.3 Constructor & Destructor Documentation

**5.630.3.1** `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 229 of file ostreambuf\_iterator.h.

**5.630.3.2** `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 233 of file ostreambuf\_iterator.h.

### 5.630.4 Member Function Documentation

**5.630.4.1** `template<typename _CharT, typename _Traits > bool  
std::ostreambuf_iterator<_CharT, _Traits >::failed () const throw ()  
[inline]`

Return true if previous [operator=\(\)](#) failed.

Definition at line 263 of file ostreambuf\_iterator.h.

**5.630.4.2** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits  
>::operator* () [inline]`

Return \*this.

Definition at line 248 of file ostreambuf\_iterator.h.

**5.630.4.3** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits  
>::operator++ () [inline]`

Return \*this.

Definition at line 258 of file ostreambuf\_iterator.h.

**5.630.4.4** `template<typename _CharT , typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator++ (int) [inline]`

Return \*this.

Definition at line 253 of file streambuf\_iterator.h.

**5.630.4.5** `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 238 of file streambuf\_iterator.h.

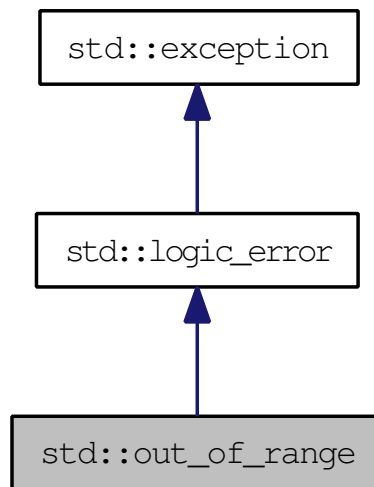
References [std::basic\\_streambuf< \\_CharT, \\_Traits >::sputc\(\)](#).

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

## 5.631 `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 ()

#### 5.631.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in [basic\\_string](#)).

Definition at line 96 of file `stdexcept`.

#### 5.631.2 Member Function Documentation

##### 5.631.2.1 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](#)



## 5.632 `std::output_iterator_tag` Struct Reference

Marking output iterators.

### 5.632.1 Detailed Description

Marking output iterators.

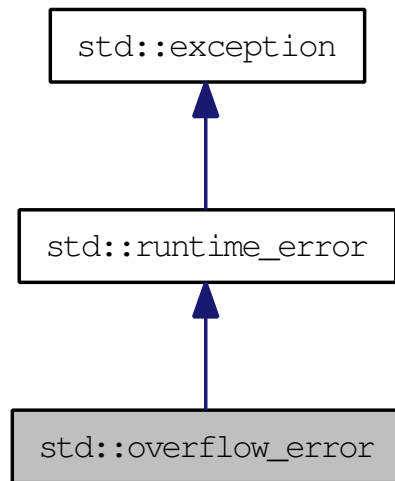
Definition at line 82 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.633 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- `overflow_error` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.633.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 133 of file `stdexcept`.

#### 5.633.2 Member Function Documentation

##### 5.633.2.1 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](#)

## 5.634 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Partial specialization of `owner_less` for [shared\\_ptr](#).

Inherits `_Sp_owner_less< shared_ptr< _Tp >, weak_ptr< _Tp > >`.

### 5.634.1 Detailed Description

`template<typename _Tp> struct std::owner_less< shared_ptr< _Tp > >`

Partial specialization of `owner_less` for [shared\\_ptr](#).

Definition at line 385 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## **5.635 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference**

Partial specialization of `owner_less` for `weak_ptr`.

Inherits `_Sp_owner_less< weak_ptr< _Tp >, shared_ptr< _Tp > >`.

### **5.635.1 Detailed Description**

```
template<typename _Tp> struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 391 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 5.636 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

`packaged_task`

### Public Types

- typedef `_Res` **result\_type**

### Public Member Functions

- **packaged\_task** (`packaged_task &&__other`)
- **packaged\_task** (`packaged_task &`)
- **packaged\_task** (`_Res(*__fn)(_ArgTypes...)`)
- `template<typename _Fn >`  
**packaged\_task** (`_Fn &&__fn`)
- `template<typename _Fn >`  
**packaged\_task** (`const _Fn &__fn`)
- **future**< `_Res` > **get\_future** ()
- **operator bool** () const
- void **operator**() (`_ArgTypes...__args`)
- `packaged_task &` **operator=** (`packaged_task &&__other`)
- `packaged_task &` **operator=** (`packaged_task &`)
- void **reset** ()
- void **swap** (`packaged_task &__other`)

### 5.636.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::packaged_task< _Res(_ArgTypes...)>`

`packaged_task`

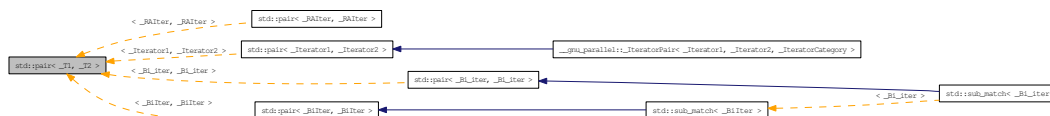
Definition at line 1183 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.637 std::pair< \_T1, \_T2 > Struct Template Reference

`pair` holds 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

- template<class `_U1`, class `_U2` >  
`pair` (`pair`< `_U1`, `_U2` > &&`_p`)
- template<class `_U1`, class `_U2` >  
`pair` (const `pair`< `_U1`, `_U2` > &`_p`)
- `pair` (`pair` &&`_p`)
- 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`)
- 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 = typename `std::enable_if`<`std::is_convertible`<`_U1`, `_T1`>::value>::type>  
`pair` (`_U1` &&`_x`, const `_T2` &`_y`)
- `pair` (const `_T1` &`_a`, const `_T2` &`_b`)
- `pair` ()
- template<class `_U1`, class `_U2` >  
`pair` & `operator=` (`pair`< `_U1`, `_U2` > &&`_p`)
- `pair` & `operator=` (`pair` &&`_p`)
- void `swap` (`pair` &`_p`)

## Public Attributes

- `_T1` [first](#)
- `_T2` [second](#)

### 5.637.1 Detailed Description

```
template<class _T1, class _T2> struct std::pair< _T1, _T2 >
```

[pair](#) holds two objects of arbitrary type.

Definition at line 71 of file `stl_pair.h`.

### 5.637.2 Member Typedef Documentation

**5.637.2.1** `template<class _T1, class _T2> typedef _T1 std::pair< _T1, _T2 >::first_type`

`first_type` is the first bound type

Definition at line 73 of file `stl_pair.h`.

**5.637.2.2** `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`second_type` is the second bound type

Definition at line 74 of file `stl_pair.h`.

### 5.637.3 Constructor & Destructor Documentation

**5.637.3.1** `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair () [inline]`

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 83 of file `stl_pair.h`.

**5.637.3.2** `template<class _T1, class _T2> 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 87 of file `stl_pair.h`.

```
5.637.3.3 template<class _T1, class _T2> template<class _U1, class _U2 >
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 118 of file `stl_pair.h`.

## 5.637.4 Member Data Documentation

```
5.637.4.1 template<class _T1, class _T2> _T1 std::pair<_T1, _T2 >::first
```

`first` is a copy of the first object

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

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set<_Key, _Compare, _Alloc >::insert()`, and `std::operator==( )`.

```
5.637.4.2 template<class _T1, class _T2> _T2 std::pair<_T1, _T2 >::second
```

`second` is a copy of the second object

Definition at line 77 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set<_Key, _Compare, _Alloc >::insert()`, and `std::operator==( )`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.638 `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

- `piecewise_constant_distribution` (const `param_type` &\_\_p)
- template<typename `_Func` >  
`piecewise_constant_distribution` (size\_t \_\_nw, `_RealType` \_\_xmin, `_RealType` \_\_xmax, `_Func` \_\_fw)
- template<typename `_Func` >  
`piecewise_constant_distribution` (`initializer_list`< `_RealType` > \_\_bl, `_Func` \_\_fw)
- template<typename `_InputIteratorB`, typename `_InputIteratorW` >  
`piecewise_constant_distribution` (`_InputIteratorB` \_\_bfirst, `_InputIteratorB` \_\_bend, `_InputIteratorW` \_\_wbegin)
- `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)
- 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::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 > &)`

### 5.638.1 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 4254 of file `random.h`.

### 5.638.2 Member Typedef Documentation

**5.638.2.1** `template<typename _RealType = double> typedef _RealType std::piecewise_constant_distribution<_RealType >::result_type`

The type of the range of the distribution.

Definition at line 4261 of file `random.h`.

### 5.638.3 Member Function Documentation

**5.638.3.1** `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 4349 of file `random.h`.

**5.638.3.2** `template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution<_RealType >::intervals () const [inline]`

Returns a `vector` of the intervals.

Definition at line 4342 of file random.h.

**5.638.3.3** `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 4378 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`.

**5.638.3.4** `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 4371 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

**5.638.3.5** `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 4364 of file random.h.

**5.638.3.6** `template<typename _RealType = double> param_type  
std::piecewise_constant_distribution< _RealType >::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 4356 of file random.h.

**5.638.3.7** `template<typename _RealType = double> void  
std::piecewise_constant_distribution<_RealType>::reset ()  
[inline]`

Resets the distribution state.

Definition at line 4335 of file `random.h`.

## 5.638.4 Friends And Related Function Documentation

**5.638.4.1** `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]`

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.

**5.638.4.2** `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](#)

## 5.639 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `piecewise_constant_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `template<typename _Func > param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)`
- `template<typename _Func > param_type (initializer_list<_RealType > __bi, _Func __fw)`
- `template<typename _InputIteratorB, typename _InputIteratorW > param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)`
- `std::vector<double > densities () const`
- `std::vector<_RealType > intervals () const`

### Friends

- class `piecewise_constant_distribution<_RealType >`

#### 5.639.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_constant_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4263 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.640 `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

- [piecewise\\_linear\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename `_Func` >  
**[piecewise\\_linear\\_distribution](#)** (size\_t \_\_nw, `_RealType` \_\_xmin, `_RealType` \_\_xmax, `_Func` \_\_fw)
- template<typename `_Func` >  
**[piecewise\\_linear\\_distribution](#)** ([initializer\\_list](#)< `_RealType` > \_\_bl, `_Func` \_\_fw)
- template<typename `_InputIteratorB`, typename `_InputIteratorW` >  
**[piecewise\\_linear\\_distribution](#)** (`_InputIteratorB` \_\_bfirst, `_InputIteratorB` \_\_bend, `_InputIteratorW` \_\_wbegin)
- `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)
- 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::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 > &)`

### 5.640.1 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 4435 of file `random.h`.

### 5.640.2 Member Typedef Documentation

**5.640.2.1** `template<typename _RealType = double> typedef _RealType  
std::piecewise_linear_distribution<_RealType >::result_type`

The type of the range of the distribution.

Definition at line 4442 of file `random.h`.

### 5.640.3 Member Function Documentation

**5.640.3.1** `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 4532 of file `random.h`.

**5.640.3.2** `template<typename _RealType = double> std::vector<_RealType>  
std::piecewise_linear_distribution<_RealType >::intervals () const  
[inline]`

Return the intervals of the distribution.

Definition at line 4524 of file random.h.

**5.640.3.3** `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 4561 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`.

**5.640.3.4** `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 4554 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

**5.640.3.5** `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 4547 of file random.h.

**5.640.3.6** `template<typename _RealType = double> param_type  
std::piecewise_linear_distribution< _RealType >::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 4539 of file random.h.

**5.640.3.7** `template<typename _RealType = double> void  
std::piecewise_linear_distribution<_RealType>::reset ()  
[inline]`

Resets the distribution state.

Definition at line 4517 of file `random.h`.

## 5.640.4 Friends And Related Function Documentation

**5.640.4.1** `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.

**5.640.4.2** `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](#)

## 5.641 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `piecewise_linear_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `template<typename _Func>`  
`param_type` (`size_t __nw`, `_RealType __xmin`, `_RealType __xmax`, `_Func __fw`)
- `template<typename _Func>`  
`param_type` (`initializer_list<_RealType> __bl`, `_Func __fw`)
- `template<typename _InputIteratorB, typename _InputIteratorW>`  
`param_type` (`_InputIteratorB __bfirst`, `_InputIteratorB __bend`, `_InputIteratorW __wbegin`)
- `std::vector<double>` `densities` () const
- `std::vector<_RealType>` `intervals` () const

### Friends

- class `piecewise_linear_distribution<_RealType>`

#### 5.641.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_linear_distribution<_RealType>::param_type`

Parameter type.

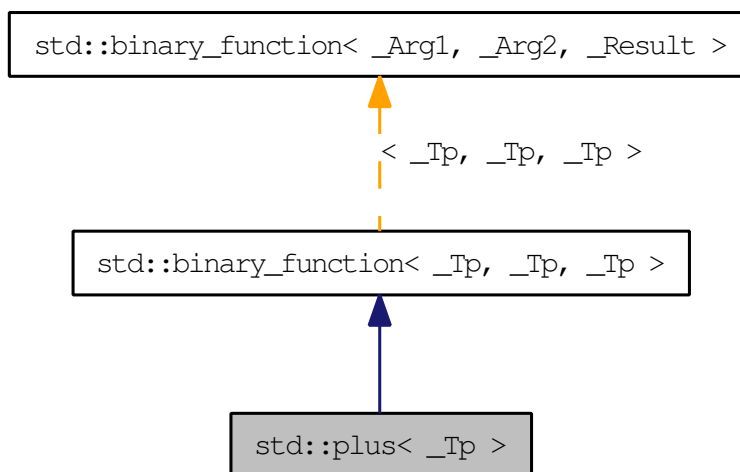
Definition at line 4444 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.642 `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`

#### 5.642.1 Detailed Description

```
template<typename _Tp> struct std::plus< _Tp >
```

One of the [math functors](#).

Definition at line 135 of file `stl_function.h`.

## 5.642.2 Member Typedef Documentation

**5.642.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.642.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.642.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::second_argument_type [inherited]`

the type of the second argument

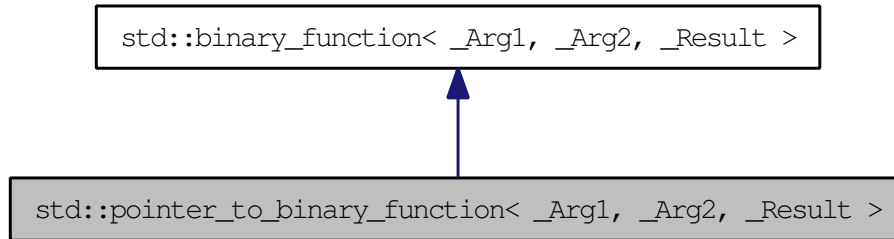
Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.643 `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)`

#### 5.643.1 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 442 of file `stl_function.h`.



## 5.643.2 Member Typedef Documentation

**5.643.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Result`  
`>::first_argument_type [inherited]`

the type of the first argument (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.643.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function<_Arg1, _Arg2, _Result`  
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.643.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

the type of the second argument

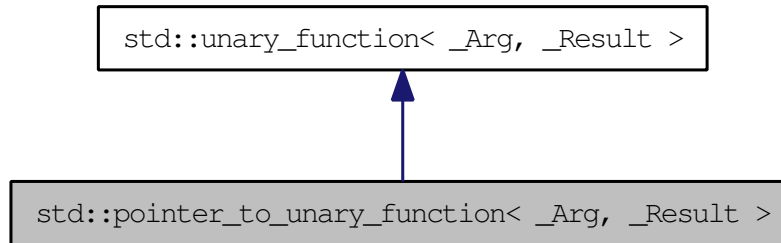
Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.644 `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)`

#### 5.644.1 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 417 of file `stl_function.h`.

## 5.644.2 Member Typedef Documentation

**5.644.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function<_Arg, _Result>::argument_type  
[inherited]`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.644.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function<_Arg, _Result>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.645 `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** (const [param\\_type](#) &\_\_p)
- **poisson\_distribution** (double \_\_mean=1.0)
- [result\\_type](#) **max** () const
- double **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** ()

### 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` , typename `_CharT` , typename `_Traits` >  
`std::basic_istream< _CharT, _Traits >` & **operator>>** (`std::basic_istream< _CharT, _Traits >` &, `std::poisson_distribution< _IntType1 >` &)

### 5.645.1 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 3493 of file `random.h`.

### 5.645.2 Member Typedef Documentation

```
5.645.2.1 template<typename _IntType = int> typedef _IntType
std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 3500 of file `random.h`.

### 5.645.3 Member Function Documentation

```
5.645.3.1 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 3583 of file `random.h`.

Referenced by `std::poisson_distribution<_IntType>::operator()`.

```
5.645.3.2 template<typename _IntType = int> double
std::poisson_distribution< _IntType >::mean () const [inline]
```

Returns the distribution parameter `mean`.

Definition at line 3554 of file `random.h`.

```
5.645.3.3 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 3576 of file `random.h`.

**5.645.3.4** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > poisson_distribution<  
_IntType >::result_type std::poisson_distribution< _IntType  
>::operator() (_UniformRandomNumberGenerator & __urng, const  
param_type & __param) [inline]`

A rejection algorithm when mean  $\geq 12$  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 1148 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::poisson_distribution< _IntType >::max()`.

**5.645.3.5** `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 3569 of file `random.h`.

**5.645.3.6** `template<typename _IntType = int> param_type  
std::poisson_distribution< _IntType >::param () const [inline]`

Returns the parameter `set` of the distribution.

Definition at line 3561 of file `random.h`.

**5.645.3.7** `template<typename _IntType = int> void std::poisson_distribution<  
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 3547 of file `random.h`.

References `std::normal_distribution< _RealType >::reset()`.

## 5.645.4 Friends And Related Function Documentation

**5.645.4.1** `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.

**5.645.4.2** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> (std::basic_istream<_CharT, _Traits > &,  
std::poisson_distribution<_IntType1 > &) [friend]`

Extracts a `poisson_distribution` random number distribution `__x` from the input stream `__is`.

**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](#)

## 5.646 `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

- class `poisson_distribution<_IntType>`

#### 5.646.1 Detailed Description

`template<typename _IntType = int> struct std::poisson_distribution<_IntType>::param_type`

Parameter type.

Definition at line 3502 of file `random.h`.

The documentation for this struct was generated from the following files:

- `random.h`
- `random.tcc`



## 5.647 `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** ([priority\\_queue](#) &&\_\_pq)
- `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`)
- **priority\_queue** (`const _Compare &__x=_Compare()`, `_Sequence &&__s=_Sequence()`)
- [priority\\_queue](#) (`const _Compare &__x`, `const _Sequence &__s`)
- `template<typename... _Args >`  
`void` **emplace** (`_Args &&...__args`)
- `bool` **empty** () `const`
- [priority\\_queue](#) & **operator=** ([priority\\_queue](#) &&\_\_pq)
- `void` **pop** ()
- `void` **push** (`value_type &&__x`)
- `void` [push](#) (`const value_type &__x`)
- `size_type` [size](#) () `const`
- `void` **swap** ([priority\\_queue](#) &\_\_pq)
- `const_reference` [top](#) () `const`

### Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

### 5.647.1 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`.

### 5.647.2 Constructor & Destructor Documentation

```
5.647.2.1 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()`.

**5.647.2.2** `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()`.

### 5.647.3 Member Function Documentation

**5.647.3.1** `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 471 of file `stl_queue.h`.

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

**5.647.3.2** `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 534 of file `stl_queue.h`.

References `std::pop_heap()`.

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

```
5.647.3.3 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 499 of file `stl_queue.h`.

References `std::push_heap()`.

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

```
5.647.3.4 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 476 of file `stl_queue.h`.

```
5.647.3.5 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 484 of file `stl_queue.h`.

**5.647 std::priority\_queue<\_Tp, \_Sequence, \_Compare > Class Template Reference**

---

**3157**

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](#)

## 5.648 `std::promise< _Res >` Class Template Reference

Primary template for [promise](#).

### Public Member Functions

- `promise` (const [promise](#) &)
- `promise` ([promise](#) &&\_\_rhs)
- `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`

### 5.648.1 Detailed Description

`template<typename _Res> class std::promise< _Res >`

Primary template for [promise](#).

Definition at line 825 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.649 `std::promise<_Res &>` Class Template Reference

Partial specialization for `promise<R&>`.

### Public Member Functions

- `promise` (const `promise` &)
- `promise` (`promise` &&\_\_rhs)
- `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`

### 5.649.1 Detailed Description

```
template<typename _Res> class std::promise<_Res &>
```

Partial specialization for `promise<R&>`.

Definition at line 915 of file `future`.

The documentation for this class was generated from the following file:

- `future`

## 5.650 `std::promise< void >` Class Template Reference

Explicit specialization for `promise<void>`.

### Public Member Functions

- `promise` (const `promise` &)
- `promise` (`promise` &&\_\_rhs)
- `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`

#### 5.650.1 Detailed Description

`template<> class std::promise< void >`

Explicit specialization for `promise<void>`.

Definition at line 993 of file `future`.

The documentation for this class was generated from the following file:

- `future`



## 5.651 `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** (`queue` &&\_\_q)
- **queue** (`_Sequence` &&\_\_c=\_Sequence())
- **queue** (`const _Sequence` &\_\_c)
- `const_reference` **back** () const
- `reference` **back** ()
- `template<typename... _Args>`  
void **emplace** (`_Args` &&...\_\_args)
- `bool` **empty** () const
- `const_reference` **front** () const
- `reference` **front** ()
- `queue` & **operator=** (`queue` &&\_\_q)
- 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 > &)`

### 5.651.1 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 89 of file `stl_queue.h`.

### 5.651.2 Constructor & Destructor Documentation

**5.651.2.1** `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 134 of file `stl_queue.h`.

### 5.651.3 Member Function Documentation

**5.651.3.1** `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 202 of file `stl_queue.h`.

**5.651.3.2** `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 191 of file `stl_queue.h`.

**5.651.3.3** `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 156 of file `stl_queue.h`.

**5.651.3.4** `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 180 of file `stl_queue.h`.

**5.651.3.5** `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 169 of file `stl_queue.h`.

**5.651.3.6** `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 244 of file `stl_queue.h`.

```
5.651.3.7 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 218 of file `stl_queue.h`.

```
5.651.3.8 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 161 of file `stl_queue.h`.

## 5.651.4 Member Data Documentation

```
5.651.4.1 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 122 of file `stl_queue.h`.

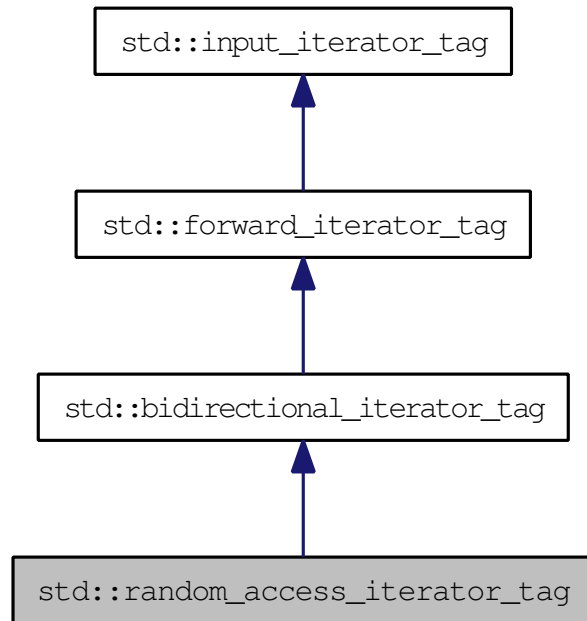
Referenced by `std::operator==(())`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.652 `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`:



### 5.652.1 Detailed Description

Random-access iterators support a superset of bidirectional [iterator](#) operations.

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](#)

## 5.653 `std::random_device` Class Reference

### Public Types

- typedef unsigned int [result\\_type](#)

### Public Member Functions

- **random\_device** (const [random\\_device](#) &)
- **random\_device** (const [std::string](#) &\_\_token="/dev/urandom")
- double **entropy** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- [result\\_type](#) **operator**() ()
- void **operator=** (const [random\\_device](#) &)

#### 5.653.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1389 of file `random.h`.

#### 5.653.2 Member Typedef Documentation

##### 5.653.2.1 `typedef unsigned int std::random_device::result_type`

The type of the generated random value.

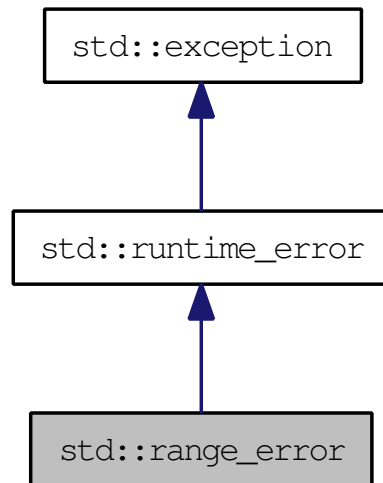
Definition at line 1393 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.654 `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 ()

#### 5.654.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 126 of file `stdexcept`.

#### 5.654.2 Member Function Documentation

##### 5.654.2.1 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](#).

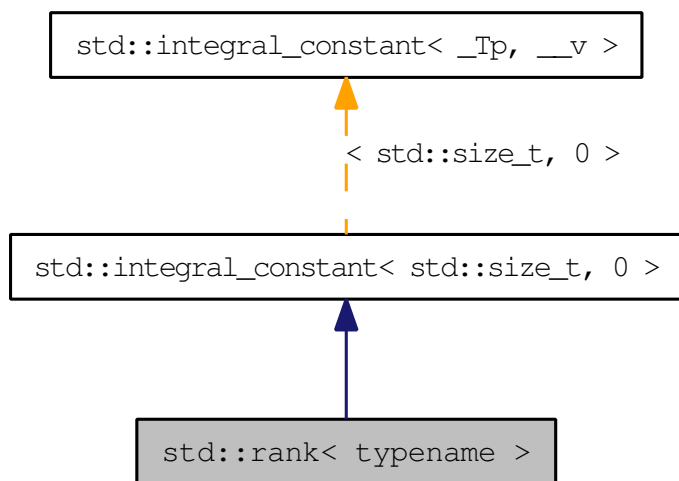
The documentation for this class was generated from the following file:

- [stdexcept](#)



## 5.655 `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**

### Static Public Attributes

- static const `std::size_t` **value**

#### 5.655.1 Detailed Description

`template<typename> struct std::rank< typename >`

[rank](#)

Definition at line 355 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.656 `std::ratio< _Num, _Den >` Struct Template Reference

Provides compile-time rational arithmetic.

### Public Member Functions

- **static\_assert** (`_Num >= __INTMAX_MAX__` && `_Den >= __INTMAX_MAX__`, "out of range")
- **static\_assert** (`_Den != 0`, "denominator cannot be zero")

### Static Public Attributes

- static const `intmax_t` **den**
- static const `intmax_t` **num**

#### 5.656.1 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 151 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.657 `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`

### 5.657.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_add< _R1, _R2 >
```

[ratio\\_add](#)

Definition at line 173 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.658 `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")

#### 5.658.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_divide< _R1, _R2 >
```

[ratio\\_divide](#)

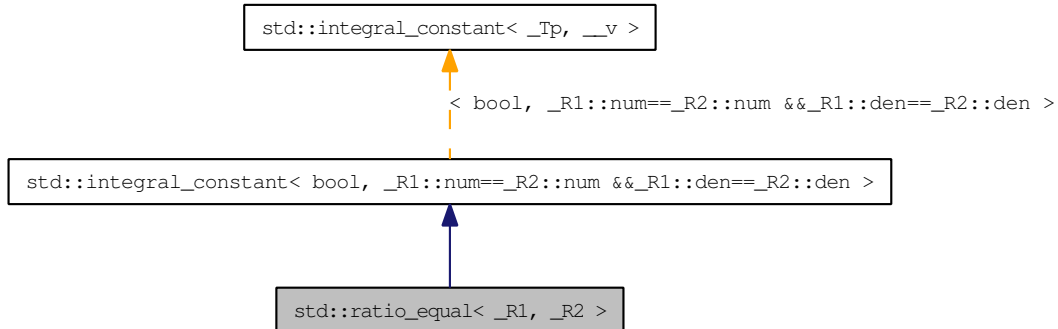
Definition at line 216 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.659 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**

### Static Public Attributes

- static const bool **value**

#### 5.659.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_equal< _R1, _R2 >
```

[ratio\\_equal](#)

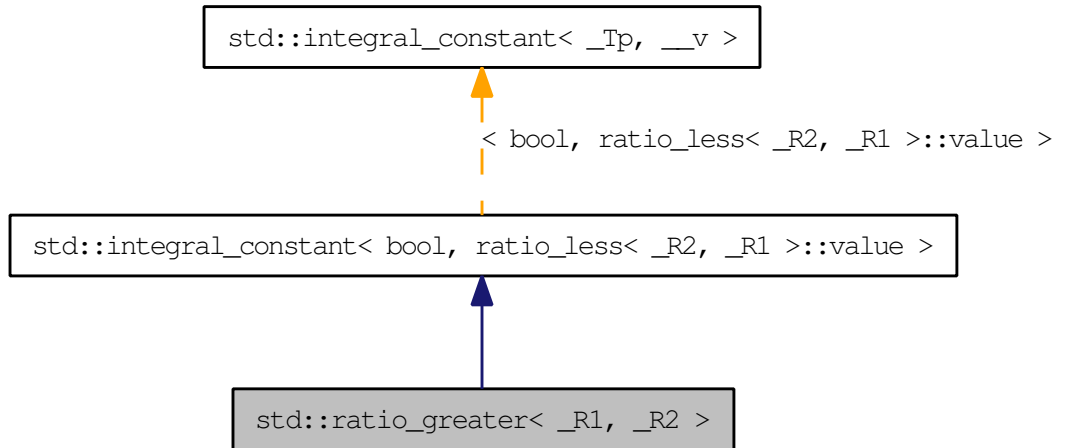
Definition at line 227 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.660 `std::ratio_greater< _R1, _R2 >` Struct Template Reference

[ratio\\_greater](#) Inheritance diagram for `std::ratio_greater< _R1, _R2 >`:



### Public Types

- typedef [integral\\_constant](#)`< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.660.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_greater< _R1, _R2 >
```

[ratio\\_greater](#)

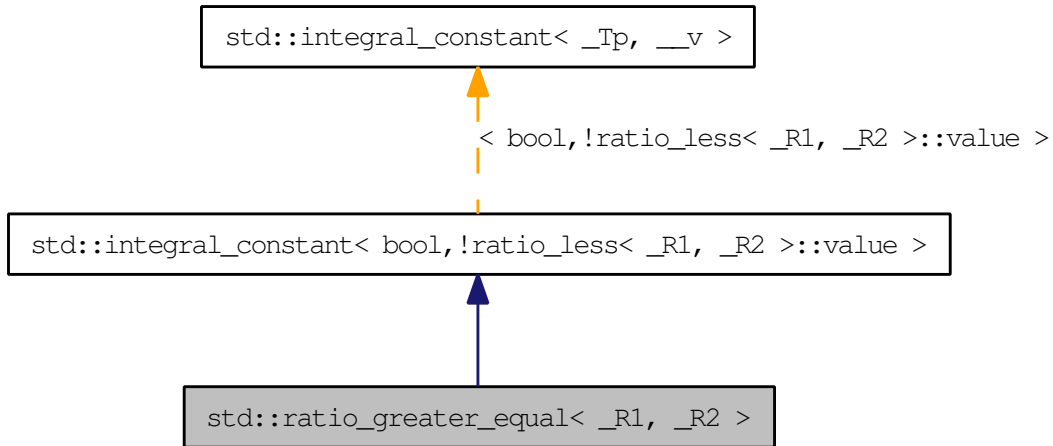
Definition at line 269 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.661 `std::ratio_greater_equal< _R1, _R2 >` Struct Template Reference

[ratio\\_greater\\_equal](#) Inheritance diagram for `std::ratio_greater_equal< _R1, _R2 >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.661.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_greater_equal< _R1,
_R2 >
```

[ratio\\_greater\\_equal](#)

Definition at line 275 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.662 `std::ratio_less< _R1, _R2 >` Struct Template Reference

[ratio\\_less](#)

Inherits `__ratio_less_impl::type< _R1, _R2 >`.

### 5.662.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_less< _R1, _R2 >
```

[ratio\\_less](#)

Definition at line 257 of file `ratio`.

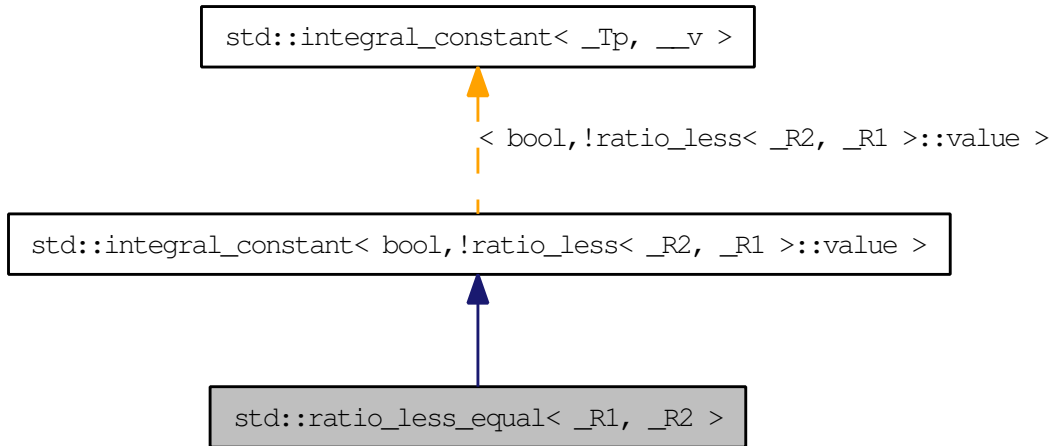
The documentation for this struct was generated from the following file:

- [ratio](#)



## 5.663 `std::ratio_less_equal<_R1, _R2>` Struct Template Reference

[ratio\\_less\\_equal](#) Inheritance diagram for `std::ratio_less_equal<_R1, _R2>`:



### Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.663.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_less_equal<_R1, _R2>
```

[ratio\\_less\\_equal](#)

Definition at line 263 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.664 `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**

### 5.664.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_multiply< _R1, _R2 >
```

[ratio\\_multiply](#)

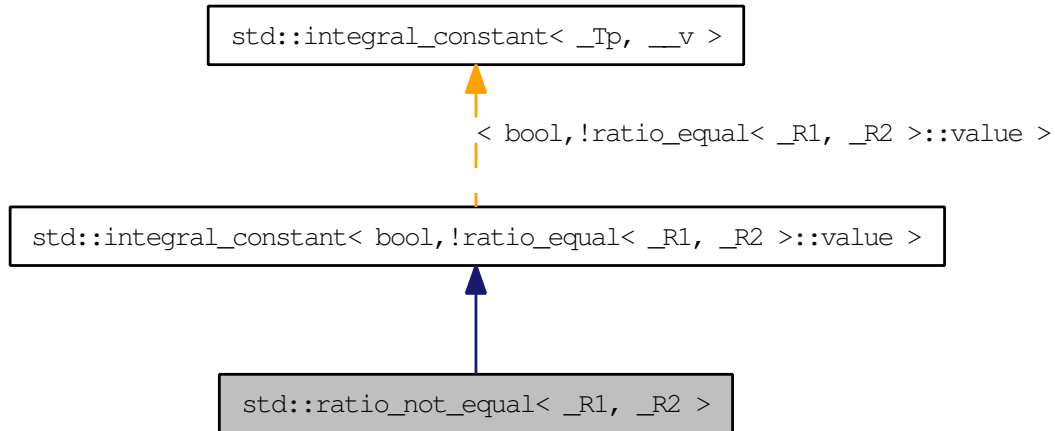
Definition at line 198 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.665 `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**

### Static Public Attributes

- static const `bool` **value**

#### 5.665.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2 >
```

[ratio\\_not\\_equal](#)

Definition at line 233 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.666 `std::ratio_subtract< _R1, _R2 >` Struct Template Reference

[ratio\\_subtract](#)

### Public Types

- typedef [ratio\\_add](#)< \_R1, [ratio](#)< \_R2::num, \_R2::den > >::type **type**

#### 5.666.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_subtract< _R1, _R2 >
```

[ratio\\_subtract](#)

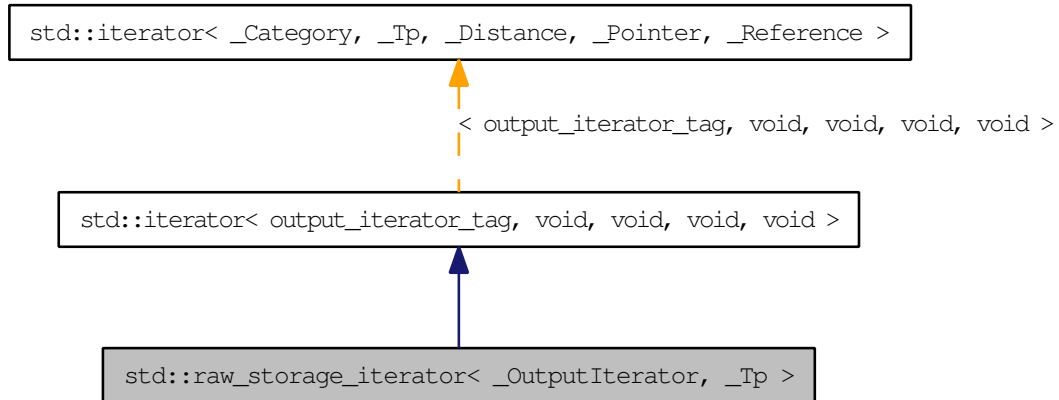
Definition at line 189 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.667 `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++` (int)
- `raw_storage_iterator< _OutputIterator, _Tp >` & `operator++` ()
- `raw_storage_iterator` & `operator=` (const `_Tp` & `__element`)

### Protected Attributes

- `_OutputIterator _M_iter`

### 5.667.1 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 67 of file `stl_raw_storage_iter.h`.

### 5.667.2 Member Typedef Documentation

**5.667.2.1** `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 115 of file `stl_iterator_base_types.h`.

**5.667.2.2** `typedef output_iterator_tag std::iterator< output_iterator_tag , void  
, void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 111 of file `stl_iterator_base_types.h`.

**5.667.2.3** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.667.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 119 of file `stl_iterator_base_types.h`.

**5.667.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 113 of file `stl_iterator_base_types.h`.

**5.667 std::raw\_storage\_iterator< \_OutputIterator, \_Tp > Class Template Reference**

---

**3183**

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 5.668 `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` ()

#### 5.668.1 Detailed Description

[recursive\\_mutex](#)

Definition at line 115 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)



## 5.669 `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` ()

### 5.669.1 Detailed Description

[recursive\\_timed\\_mutex](#)

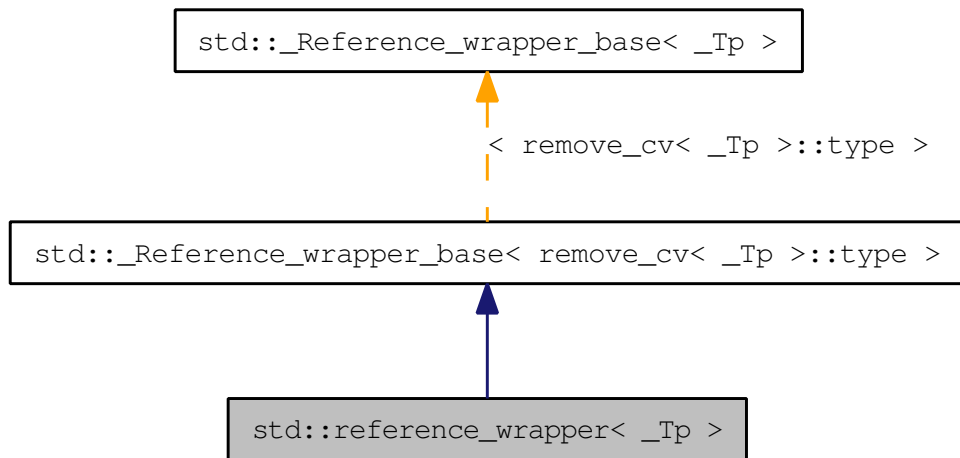
Definition at line 271 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.670 `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` (const [reference\\_wrapper](#)< `_Tp` > &\_\_inref)
- `reference_wrapper` (`_Tp` &&)
- `reference_wrapper` (`_Tp` &\_\_indata)
- `_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)

### **5.670.1 Detailed Description**

**template<typename \_Tp> class std::reference\_wrapper<\_Tp >**

Primary class template for [reference\\_wrapper](#).

Definition at line 392 of file functional.

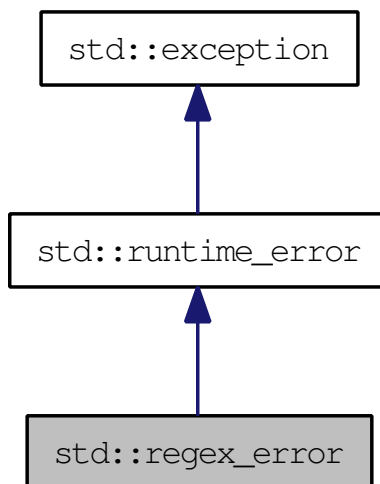
The documentation for this class was generated from the following file:

- [functional](#)

## 5.671 `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

#### 5.671.1 Detailed Description

A regular expression [exception](#) class.

The regular expression library throws objects of this class on error.

Definition at line 395 of file `tr1_impl/regex`.

## 5.671.2 Constructor & Destructor Documentation

### 5.671.2.1 `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 405 of file `tr1_impl/regex`.

## 5.671.3 Member Function Documentation

### 5.671.3.1 `regex_constants::error_type std::regex_error::code () const` [`inline`]

Gets the regex error code.

#### Returns:

the regex error code.

Definition at line 415 of file `tr1_impl/regex`.

### 5.671.3.2 `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:

- [tr1\\_impl/regex](#)

## 5.672 `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` (const `regex_iterator` &\_\_rhs)
- `regex_iterator` (`_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- `regex_iterator` ()
- `bool operator!=` (const `regex_iterator` &\_\_rhs)
- const `value_type` & `operator*` ()
- `regex_iterator operator++` (int)
- `regex_iterator & operator++` ()
- const `value_type` \* `operator->` ()
- `regex_iterator & operator=` (const `regex_iterator` &\_\_rhs)
- `bool operator==` (const `regex_iterator` &\_\_rhs)

#### 5.672.1 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 2431 of file `tr1_impl/regex`.

## 5.672.2 Constructor & Destructor Documentation

**5.672.2.1** `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.

Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.672.2.2** `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.

Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.672.2.3** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**5.672.3 Member Function Documentation**

**5.672.3.1** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**5.672.3.2** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**5.672.3.3** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.



**5.672 std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits > Class Template Reference** 3193

---

**5.672.3.4** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.672.3.5** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.672.3.6** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.672.3.7** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

The documentation for this class was generated from the following file:

- [tr1\\_impl/regex](#)

## 5.673 `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` (`const regex_token_iterator &__rhs`)
- `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` (`_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`)
- `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`)
- `regex_token_iterator` ()
- `bool operator!=` (`const regex_token_iterator &__rhs`)
- `const value_type & operator*` ()
- `regex_token_iterator operator++` (`int`)
- `regex_token_iterator & operator++` ()
- `const value_type * operator->` ()
- `regex_token_iterator & operator=` (`const regex_token_iterator &__rhs`)
- `bool operator==` (`const regex_token_iterator &__rhs`)

#### 5.673.1 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::tr1::sub_match` object.

Definition at line 2546 of file `tr1_impl/regex`.

## 5.673.2 Constructor & Destructor Documentation

**5.673.2.1** `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.

**5.673.2.2** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.673.2.3** `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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**5.673.2.4** `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) [inline]`

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.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

**5.673.2.5** `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.

**5.673.3 Member Function Documentation**

**5.673.3.1** `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.

**5.673.3.2** `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.

**5.673.3.3** `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.

**5.673.3.4** `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.

**5.673.3.5** `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.

**5.673.3.6** `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.

**5.673.3.7** `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:

- [tr1\\_impl/regex](#)



## 5.674 `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) 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`

### 5.674.1 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 [tr1](#) section [7.2].

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 434 of file `tr1_impl/regex`.

### 5.674.2 Constructor & Destructor Documentation

**5.674.2.1** `template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits () [inline]`

Constructs a default traits object.

Definition at line 446 of file `tr1_impl/regex`.

### 5.674.3 Member Function Documentation

**5.674.3.1** `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 646 of file `tr1_impl/regex`.

**5.674.3.2** `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 635 of file tr1\_impl/regex.

References std::swap().

**5.674.3.3** `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 460 of file tr1\_impl/regex.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::length().

**5.674.3.4** `template<typename _Ch_type > template<typename _Fwd_iter > char_class_type std::regex_traits< _Ch_type >::lookup_classname (_Fwd_iter __first, _Fwd_iter __last) 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.

**Returns:**

an unspecified value that represents the character classification named by the character sequence designated by the [iterator](#) range [first, last). 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.

Referenced by `std::regex_traits<_Ch_type>::isctype()`.

**5.674.3.5** `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.

```
5.674.3.6 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 513 of file tr1\_impl/regex.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::data\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::size\(\)](#), [std::collate< \\_CharT >::transform\(\)](#), and [std::use\\_facet\(\)](#).

```
5.674.3.7 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]
```

Dunno.

**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. WTF??

**Todo**

Implement this function.

**5.674.3.8** `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 471 of file `tr1_impl/regex`.

**5.674.3.9** `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 484 of file `tr1_impl/regex`.

References `std::tolower()`, and `std::use_facet()`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/regex](#)

## 5.675 `std::remove_all_extents< _Tp >` Struct Template Reference

[remove\\_all\\_extents](#)

### Public Types

- `typedef _Tp type`

### 5.675.1 Detailed Description

```
template<typename _Tp> struct std::remove_all_extents< _Tp >
```

[remove\\_all\\_extents](#)

Definition at line 459 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.676 `std::remove_const< _Tp >` Struct Template Reference

[remove\\_const](#)

### Public Types

- `typedef _Tp type`

#### 5.676.1 Detailed Description

`template<typename _Tp> struct std::remove_const< _Tp >`

[remove\\_const](#)

Definition at line 400 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)



## 5.677 `std::remove_cv< _Tp >` Struct Template Reference

[remove\\_cv](#)

### Public Types

- typedef [remove\\_const](#)< typename [remove\\_volatile](#)< \_Tp >::type >::type **type**

### 5.677.1 Detailed Description

```
template<typename _Tp> struct std::remove_cv< _Tp >
```

[remove\\_cv](#)

Definition at line 418 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.678 `std::remove_extent< _Tp >` Struct Template Reference

[remove\\_extent](#)

### Public Types

- `typedef _Tp type`

#### 5.678.1 Detailed Description

`template<typename _Tp> struct std::remove_extent< _Tp >`

[remove\\_extent](#)

Definition at line 446 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.679 `std::remove_pointer< _Tp >` Struct Template Reference

[remove\\_pointer](#)

Inherits `std::__remove_pointer_helper< _Tp, remove_cv< _Tp >::type >`.

### Public Types

- `typedef _Tp type`

### 5.679.1 Detailed Description

```
template<typename _Tp> struct std::remove_pointer< _Tp >
```

[remove\\_pointer](#)

Definition at line 482 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.680 `std::remove_reference< _Tp >` Struct Template Reference

[remove\\_reference](#)

### Public Types

- `typedef _Tp type`

#### 5.680.1 Detailed Description

`template<typename _Tp> struct std::remove_reference< _Tp >`

[remove\\_reference](#)

Definition at line 98 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.681 `std::remove_volatile< _Tp >` Struct Template Reference

[remove\\_volatile](#)

### Public Types

- `typedef _Tp type`

#### 5.681.1 Detailed Description

`template<typename _Tp> struct std::remove_volatile< _Tp >`

[remove\\_volatile](#)

Definition at line 409 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.682 `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

- `template<typename _Iter >`  
[reverse\\_iterator](#) (const [reverse\\_iterator](#)< \_Iter > &\_\_x)
- [reverse\\_iterator](#) (const [reverse\\_iterator](#) &\_\_x)
- [reverse\\_iterator](#) (iterator\_type \_\_x)
- [reverse\\_iterator](#) ()
- iterator\_type [base](#) () const
- [reference](#) [operator\\*](#) () const
- [reverse\\_iterator](#) [operator+](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) [operator++](#) (int)
- [reverse\\_iterator](#) & [operator++](#) ()
- [reverse\\_iterator](#) & [operator+=](#) (difference\_type \_\_n)
- [reverse\\_iterator](#) [operator-](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) [operator--](#) (int)
- [reverse\\_iterator](#) & [operator--](#) ()
- [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`

### 5.682.1 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 95 of file `stl_iterator.h`.

### 5.682.2 Member Typedef Documentation

**5.682.2.1 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 109 of file `stl_iterator.h`.

**5.682.2.2 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 111 of file `stl_iterator_base_types.h`.

**5.682.2.3** `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 110 of file `stl_iterator.h`.

**5.682.2.4** `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 111 of file `stl_iterator.h`.

**5.682.2.5** `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 113 of file `stl_iterator_base_types.h`.

## 5.682.3 Constructor & Destructor Documentation

**5.682.3.1** `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 119 of file `stl_iterator.h`.



**5.682.3.2** `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 125 of file `stl_iterator.h`.

**5.682.3.3** `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator (const reverse_iterator< _Iterator > & __x) [inline]`

The copy constructor is normal.

Definition at line 130 of file `stl_iterator.h`.

**5.682.3.4** `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 138 of file `stl_iterator.h`.

## 5.682.4 Member Function Documentation

**5.682.4.1** `template<typename _Iterator> iterator_type std::reverse_iterator< _Iterator >::base () const [inline]`

### Returns:

`current`, the iterator used for underlying work.

Definition at line 145 of file `stl_iterator.h`.

Referenced by `std::operator==()`.

**5.682.4.2** `template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator* () const [inline]`

### Returns:

TODO

### Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 154 of file stl\_iterator.h.

**5.682.4.3** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator+ (difference_type __n)  
const [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 225 of file stl\_iterator.h.

**5.682.4.4** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator++ (int) [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 187 of file stl\_iterator.h.

**5.682.4.5** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator++ () [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 175 of file stl\_iterator.h.

**5.682.4.6** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator+=( difference_type __n)  
[inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 234 of file stl\_iterator.h.

**5.682.4.7** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator- (difference_type __n)  
const [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 246 of file stl\_iterator.h.

**5.682.4.8** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator-- (int) [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 212 of file stl\_iterator.h.

**5.682.4.9** `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-- () [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 200 of file stl\_iterator.h.

**5.682.4.10** `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-= (difference_type __n) [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 255 of file stl\_iterator.h.

**5.682.4.11** `template<typename _Iterator> pointer std::reverse_iterator< _Iterator >::operator-> () const [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 166 of file stl\_iterator.h.

**5.682.4.12** `template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator[] (difference_type __n) const [inline]`

**Returns:**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

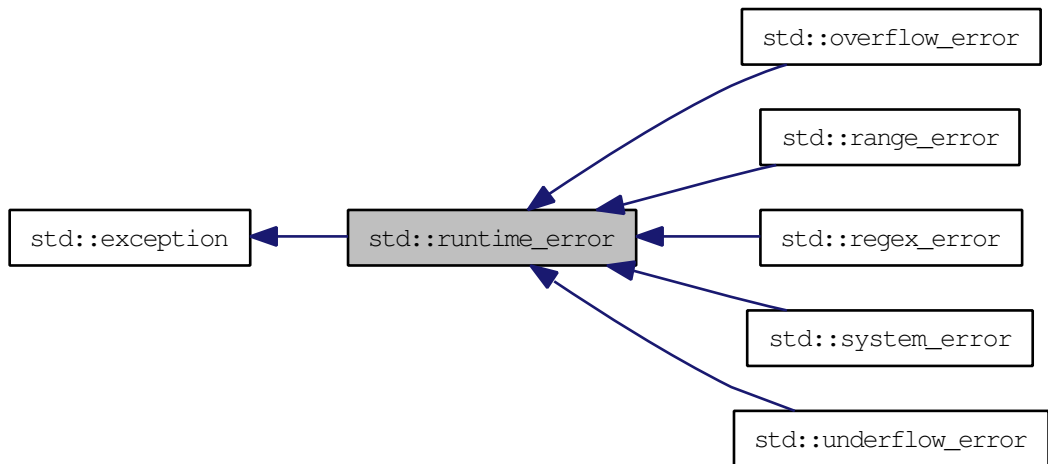
Definition at line 267 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.683 `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 ()

#### 5.683.1 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 107 of file `stdexcept`.

#### 5.683.2 Constructor & Destructor Documentation

##### 5.683.2.1 `std::runtime_error::runtime_error` (const [string](#) & \_\_arg) [explicit]

Takes a character string describing the error.

### 5.683.3 Member Function Documentation

#### 5.683.3.1 `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](#)

## 5.684 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

- `template<typename _InputIterator >`  
`seed_seq` (`_InputIterator __begin`, `_InputIterator __end`)
- `template<typename _IntType >`  
`seed_seq` (`std::initializer_list<_IntType > il`)
- `seed_seq` ()
- `template<typename _RandomAccessIterator >`  
`void generate` (`_RandomAccessIterator __begin`, `_RandomAccessIterator __end`)
- `template<typename OutputIterator >`  
`void param` (`OutputIterator __dest`) `const`
- `size_t size` () `const`

### 5.684.1 Detailed Description

The [seed\\_seq](#) class generates sequences of seeds for random number generators.

Definition at line 4625 of file `random.h`.

### 5.684.2 Member Typedef Documentation

#### 5.684.2.1 typedef `uint_least32_t` `std::seed_seq::result_type`

The type of the seed vales.

Definition at line 4630 of file `random.h`.

### 5.684.3 Constructor & Destructor Documentation

#### 5.684.3.1 `std::seed_seq::seed_seq` () [`inline`]

Default constructor.



Definition at line 4633 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.685 `std::set< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of unique keys, which can be retrieved in logarithmic time.

### Public Member Functions

- `set` (`initializer_list< value_type > __l`, `const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `set` (`set &&__x`)
- `set` (`const set &__x`)
- `template<typename _InputIterator >`  
`set` (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
`set` (`_InputIterator __first`, `_InputIterator __last`)
- `set` (`const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `set` ()
- `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`
- `iterator erase` (`iterator __first`, `iterator __last`)
- `size_type erase` (`const key_type &__x`)
- `iterator erase` (`iterator __position`)
- `allocator_type get_allocator` () `const`
- `void insert` (`initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `iterator insert` (`iterator __position`, `const value_type &__x`)
- `std::pair< iterator, bool >` `insert` (`const value_type &__x`)
- `key_compare key_comp` () `const`
- `size_type max_size` () `const`
- `set & operator=` (`initializer_list< value_type > __l`)
- `set & operator=` (`set &&__x`)
- `set & operator=` (`const set &__x`)

- [reverse\\_iterator rbegin](#) () const
- [reverse\\_iterator rend](#) () const
- [size\\_type size](#) () const
- void [swap](#) ([set](#) &\_\_x)
- [value\\_compare value\\_comp](#) () 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 >`  
`&)`
- `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`
- `std::pair< const\_iterator, const\_iterator > equal\_range (const key\_type &__x)`  
`const`
- `std::pair< iterator, iterator > equal\_range (const key\_type &__x)`
- `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)`

### 5.685.1 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 87 of file stl\_set.h.

### 5.685.2 Member Typedef Documentation

**5.685.2.1** `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 104 of file stl\_set.h.

**5.685.2.2** `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`

Public typedefs.

Definition at line 125 of file stl\_set.h.

**5.685.2.3** `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`

Public typedefs.

Definition at line 118 of file stl\_set.h.

**5.685.2.4** `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`

Public typedefs.

Definition at line 120 of file stl\_set.h.

**5.685.2.5** `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`

Public typedefs.

Definition at line 127 of file stl\_set.h.

**5.685.2.6** `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`

Public typedefs.

Definition at line 129 of file stl\_set.h.

**5.685.2.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc  
>::iterator`

Public typedefs.

Definition at line 124 of file stl\_set.h.

**5.685.2.8** `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 102 of file `stl_set.h`.

**5.685.2.9** `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 100 of file `stl_set.h`.

**5.685.2.10** `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 117 of file `stl_set.h`.

**5.685.2.11** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::reference std::set< _Key, _Compare, _Alloc  
>::reference`

Public typedefs.

Definition at line 119 of file `stl_set.h`.

**5.685.2.12** `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`

Public typedefs.

Definition at line 126 of file `stl_set.h`.

**5.685.2.13** `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`

Public typedefs.

Definition at line 128 of file stl\_set.h.

**5.685.2.14** `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 103 of file stl\_set.h.

**5.685.2.15** `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 101 of file stl\_set.h.

### 5.685.3 Constructor & Destructor Documentation

**5.685.3.1** `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 136 of file stl\_set.h.

**5.685.3.2** `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 145 of file `stl_set.h`.

**5.685.3.3** `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 159 of file `stl_set.h`.

**5.685.3.4** `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 175 of file `stl_set.h`.



```
5.685.3.5 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 188 of file stl\_set.h.

```
5.685.3.6 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 199 of file stl\_set.h.

```
5.685.3.7 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 *N*log*N* otherwise (where *N* is *l.size()*).

Definition at line 212 of file stl\_set.h.

## 5.685.4 Member Function Documentation

**5.685.4.1** `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 292 of file `stl_set.h`.

**5.685.4.2** `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 329 of file `stl_set.h`.

**5.685.4.3** `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 338 of file `stl_set.h`.

**5.685.4.4** `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 552 of file `stl_set.h`.

**5.685.4.5** `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 566 of file `stl_set.h`.

**5.685.4.6** `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 347 of file `stl_set.h`.

**5.685.4.7** `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 356 of file `stl_set.h`.

**5.685.4.8** `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 362 of file `stl_set.h`.

**5.685.4.9** `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 301 of file `stl_set.h`.

**5.685.4.10** `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]`

Public typedefs.

Definition at line 650 of file `stl_set.h`.

**5.685.4.11** `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 646 of file `stl_set.h`.

**5.685.4.12** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set<_Key, _Compare, _Alloc >::erase (iterator __first, 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 526 of file stl\_set.h.

```
5.685.4.13 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 507 of file stl\_set.h.

```
5.685.4.14 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::erase (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 478 of file `stl_set.h`.

```
5.685.4.15 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]
```

Public typedefs.

Definition at line 588 of file `stl_set.h`.

```
5.685.4.16 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 584 of file `stl_set.h`.

```
5.685.4.17 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 283 of file `stl_set.h`.

**5.685.4.18** `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 458 of file stl\_set.h.

References std::set< \_Key, \_Compare, \_Alloc >::insert().

Referenced by std::set< \_Key, \_Compare, \_Alloc >::insert().

**5.685.4.19** `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 446 of file stl\_set.h.

**5.685.4.20** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,  
_Compare, _Alloc >::insert (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>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 432 of file `stl_set.h`.

```
5.685.4.21 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 405 of file `stl_set.h`.

References `std::pair<_T1, _T2 >::first`, and `std::pair<_T1, _T2 >::second`.

```
5.685.4.22 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> key_compare std::set<
_Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

Definition at line 275 of file `stl_set.h`.



**5.685.4.23** `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]`

Public typedefs.

Definition at line 609 of file stl\_set.h.

**5.685.4.24** `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 605 of file stl\_set.h.

**5.685.4.25** `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 372 of file stl\_set.h.

**5.685.4.26** `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 263 of file `stl_set.h`.

```
5.685.4.27 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 242 of file `stl_set.h`.

References `std::swap()`.

```
5.685.4.28 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 227 of file `stl_set.h`.

**5.685.4.29** `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 310 of file `stl_set.h`.

**5.685.4.30** `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 319 of file `stl_set.h`.

**5.685.4.31** `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 367 of file `stl_set.h`.

**5.685.4.32** `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 387 of file `stl_set.h`.

Referenced by `std::swap()`.

```
5.685.4.33 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]
```

Public typedefs.

Definition at line 625 of file stl\_set.h.

```
5.685.4.34 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 621 of file stl\_set.h.

```
5.685.4.35 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.

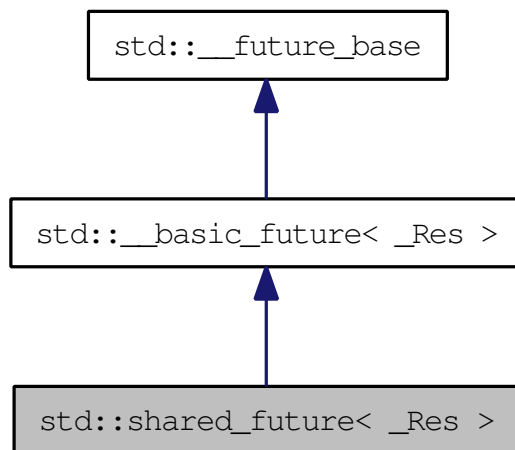
Definition at line 279 of file stl\_set.h.

The documentation for this class was generated from the following file:

- [stl\\_set.h](#)

## 5.686 `std::shared_future< _Res >` Class Template Reference

Primary template for `shared_future`. Inheritance diagram for `std::shared_future< _Res >`:



### Public Member Functions

- `shared_future` (`shared_future` &&\_sf)
- `shared_future` (`future< _Res >` &&\_uf)
- `shared_future` (const `shared_future` &\_sf)
- const `_Res` & `get` ()
- `shared_future` & `operator=` (`shared_future` &&\_sf)
- `shared_future` & `operator=` (const `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

### 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 \(\\_\\_basic\\_future &\\_\\_that\)](#)

### 5.686.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res >`

Primary template for [shared\\_future](#).

Definition at line 683 of file future.

### 5.686.2 Constructor & Destructor Documentation

**5.686.2.1** `template<typename _Res > std::shared_future< _Res >::shared_future (const shared_future< _Res > & __sf) [inline]`

Copy constructor.

Definition at line 691 of file future.

**5.686.2.2** `template<typename _Res > std::shared_future< _Res >::shared_future (future< _Res > && __uf) [inline]`

Construct from a [future](#) rvalue.

Definition at line 694 of file future.

**5.686.2.3** `template<typename _Res > std::shared_future< _Res >::shared_future (shared_future< _Res > && __sf) [inline]`

Construct from a [shared\\_future](#) rvalue.

Definition at line 699 of file future.

### 5.686.3 Member Function Documentation

**5.686.3.1** `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 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

**5.686.3.2 template<typename \_Res > const \_Res& std::shared\_future< \_Res >::get () [inline]**

Retrieving the value.

Definition at line 717 of file `future`.

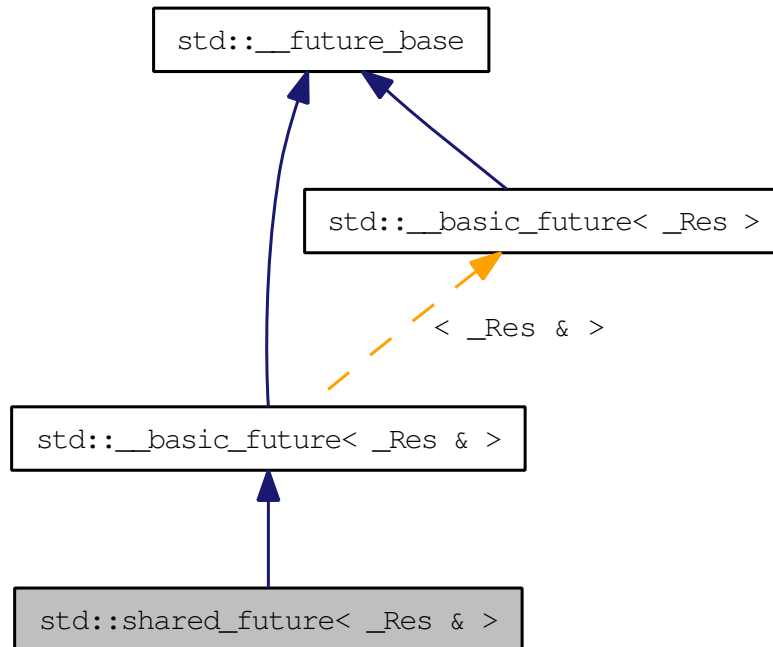
References `std::_basic_future< _Res >::_M_get_result()`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.687 `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` (`shared_future` &&\_sf)
- `shared_future` (`future<_Res &>` &&\_uf)
- `shared_future` (`const shared_future` &\_sf)
- `_Res &` `get` ()
- `shared_future` & `operator=` (`shared_future` &&\_sf)
- `shared_future` & `operator=` (`const 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`



## 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)`

### 5.687.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res & >`

Partial specialization for `shared_future<R&>`.

Definition at line 727 of file future.

### 5.687.2 Constructor & Destructor Documentation

**5.687.2.1** `template<typename _Res > std::shared_future< _Res & >::shared_future (const shared_future< _Res & > & __sf) [inline]`

Copy constructor.

Definition at line 735 of file future.

**5.687.2.2** `template<typename _Res > std::shared_future< _Res & >::shared_future (future< _Res & > && __uf) [inline]`

Construct from a `future` rvalue.

Definition at line 738 of file future.

**5.687.2.3** `template<typename _Res > std::shared_future< _Res & >::shared_future (shared_future< _Res & > && __sf) [inline]`

Construct from a `shared_future` rvalue.

Definition at line 743 of file future.

### 5.687.3 Member Function Documentation

#### 5.687.3.1 `__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 513 of file future.

References `std::rethrow_exception()`.

#### 5.687.3.2 `template<typename _Res > _Res& std::shared_future<_Res & >::get () [inline]`

Retrieving the value.

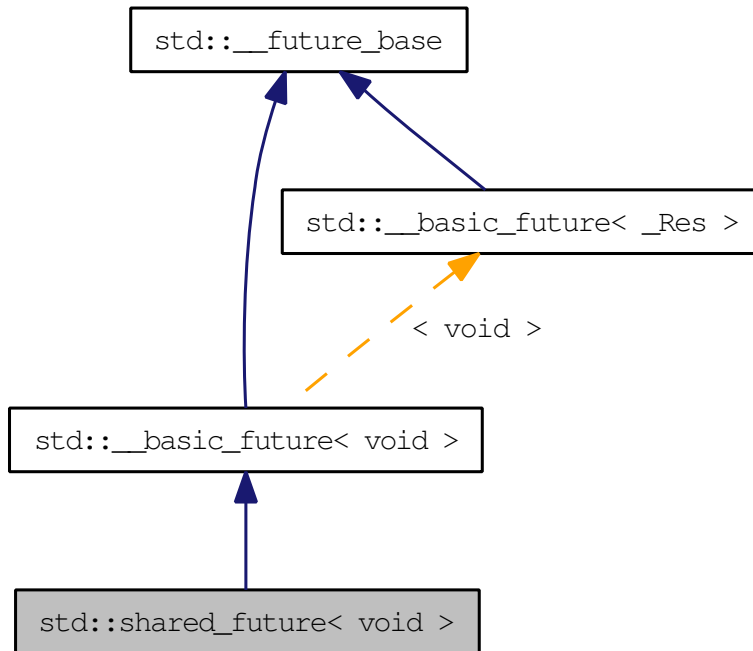
Definition at line 761 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.688 `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` (`shared_future` &&\_sf)
- `shared_future` (`future< void >` &&\_uf)
- `shared_future` (const `shared_future` &\_sf)
- void `get` ()
- `shared_future` & `operator=` (`shared_future` &&\_sf)
- `shared_future` & `operator=` (const `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

## 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`)

### 5.688.1 Detailed Description

`template<> class std::shared_future< void >`

Explicit specialization for `shared_future<void>`.

Definition at line 766 of file future.

### 5.688.2 Constructor & Destructor Documentation

**5.688.2.1** `std::shared_future< void >::shared_future (const shared_future< void > & __sf)` [`inline`]

Copy constructor.

Definition at line 774 of file future.

**5.688.2.2** `std::shared_future< void >::shared_future (future< void > && __uf)` [`inline`]

Construct from a `future` rvalue.

Definition at line 777 of file future.

**5.688.2.3** `std::shared_future< void >::shared_future (shared_future< void > && __sf)` [`inline`]

Construct from a `shared_future` rvalue.

Definition at line 782 of file future.

### 5.688.3 Member Function Documentation

#### 5.688.3.1 `__result_type std::__basic_future< void >::_M_get_result ()` `[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

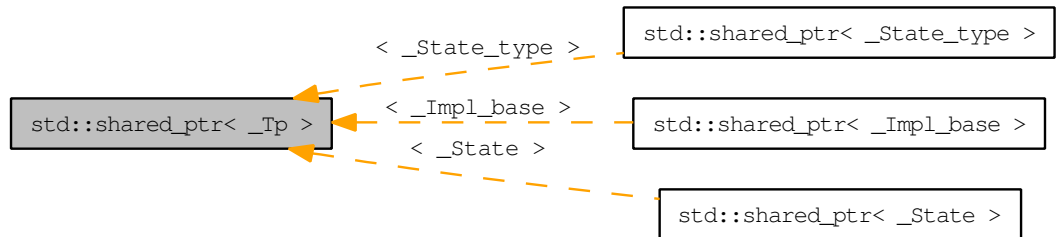
Definition at line 513 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.689 `std::shared_ptr< _Tp >` Class Template Reference

A smart pointer with reference-counted copy semantics. Inheritance diagram for `std::shared_ptr< _Tp >`:



### Public Member Functions

- `template<typename _Tp1, typename _Del >`  
**shared\_ptr** (`std::unique_ptr< _Tp1, _Del > &&__r`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`std::auto_ptr< _Tp1 > &&__r`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`const weak_ptr< _Tp1 > &__r`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`shared_ptr< _Tp1 > &&__r`)
- `shared_ptr` (`shared_ptr &&__r`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`const shared_ptr< _Tp1 > &__r`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`const shared_ptr< _Tp1 > &__r, _Tp * __p`)
- `template<typename _Tp1, typename _Deleter, typename _Alloc >`  
**shared\_ptr** (`_Tp1 * __p, _Deleter __d, const _Alloc &__a`)
- `template<typename _Tp1, typename _Deleter >`  
**shared\_ptr** (`_Tp1 * __p, _Deleter __d`)
- `template<typename _Tp1 >`  
**shared\_ptr** (`_Tp1 * __p`)
- `shared_ptr` (`()`)
- `template<typename _Tp1, typename _Del >`  
**shared\_ptr & operator=** (`std::unique_ptr< _Tp1, _Del > &&__r`)
- `template<class _Tp1 >`  
**shared\_ptr & operator=** (`shared_ptr< _Tp1 > &&__r`)
- `shared_ptr & operator= (shared_ptr &&__r)`

- `template<typename _Tp1 >`  
`shared_ptr & operator= (std::auto_ptr<_Tp1 > &&_r)`
- `template<typename _Tp1 >`  
`shared_ptr & operator= (const shared_ptr<_Tp1 > &_r)`

## Friends

- `template<typename _Tp1 , typename _Alloc , typename... _Args >`  
`shared_ptr<_Tp1 > allocate_shared (_Alloc __a, _Args &&...__args)`

### 5.689.1 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 91 of file `shared_ptr.h`.

### 5.689.2 Constructor & Destructor Documentation

**5.689.2.1** `template<typename _Tp> std::shared_ptr<_Tp >::shared_ptr ()`  
[`inline`]

Construct an empty `shared_ptr`.

**Postcondition:**

`use_count()==0 && get()==0`

Definition at line 98 of file `shared_ptr.h`.

**5.689.2.2** `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 107 of file `shared_ptr.h`.

**5.689.2.3** `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 123 of file `shared_ptr.h`.

**5.689.2.4** `template<typename _Tp> template<typename _Tp1, typename  
_Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr  
( _Tp1 * __p, _Deleter __d, const _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 141 of file `shared_ptr.h`.

```
5.689.2.5 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 163 of file `shared_ptr.h`.

```
5.689.2.6 template<typename _Tp> template<typename _Tp1 >
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 174 of file `shared_ptr.h`.

**5.689.2.7** `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 181 of file `shared_ptr.h`.

**5.689.2.8** `template<typename _Tp> template<typename _Tp1 > 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 190 of file `shared_ptr.h`.

**5.689.2.9** `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 202 of file shared\_ptr.h.

### 5.689.3 Friends And Related Function Documentation

**5.689.3.1** `template<typename _Tp> template<typename _Tp1, typename  
_Alloc, typename... _Args> shared_ptr<_Tp1> allocate_shared  
(_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 453 of file shared\_ptr.h.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.690 `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

- `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](#) (`result_type __s`)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine &&__rne`)
- [shuffle\\_order\\_engine](#) (`const _RandomNumberEngine &__rne`)
- [shuffle\\_order\\_engine](#) ()
- `const _RandomNumberEngine &` [base](#) () `const`
- void [discard](#) (`unsigned long long __z`)
- `result_type` [max](#) () `const`
- `result_type` [min](#) () `const`
- `result_type` [operator](#)() ()
- `template<typename _Sseq >` void [seed](#) (`_Sseq &__q`)
- void [seed](#) (`result_type __s`)
- void [seed](#) ()

### Static Public Attributes

- static `const 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` > &)

## 5.690 `std::shuffle_order_engine<_RandomNumberEngine, __k>` Class

### Template Reference

3261

- `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> &)`

### 5.690.1 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 1123 of file `random.h`.

### 5.690.2 Member Typedef Documentation

**5.690.2.1** `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 1130 of file `random.h`.

### 5.690.3 Constructor & Destructor Documentation

**5.690.3.1** `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 1139 of file `random.h`.

**5.690.3.2** `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:**

*rng* An existing (base class) engine object.

Definition at line 1150 of file `random.h`.

**5.690.3.3** `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:**

*rng* An existing (base class) engine object.

Definition at line 1161 of file `random.h`.

**5.690.3.4** `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 1172 of file `random.h`.

---

```
5.690.3.5 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 1186 of file `random.h`.

## 5.690.4 Member Function Documentation

```
5.690.4.1 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 1229 of file `random.h`.

```
5.690.4.2 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.

**Todo**

Look for a faster way to do discard.

Definition at line 1256 of file `random.h`.

```
5.690.4.3 template<typename _RandomNumberEngine, size_t __k>
 result_type std::shuffle_order_engine<_RandomNumberEngine, __k
 >::max () const [inline]
```

Gets the maximum value in the generated random number range.

**Todo**

This should be `constexpr`.

Definition at line 1247 of file random.h.

**5.690.4.4** `template<typename _RandomNumberEngine, size_t __k>  
result_type std::shuffle_order_engine< _RandomNumberEngine, __k  
>::min () const [inline]`

Gets the minimum value in the generated random number range.

#### Todo

This should be constexpr.

Definition at line 1238 of file random.h.

**5.690.4.5** `template<typename _RandomNumberEngine , size_t __k>  
shuffle_order_engine< _RandomNumberEngine, __k >::result_type  
std::shuffle_order_engine< _RandomNumberEngine, __k  
>::operator() () [inline]`

Gets the next value in the generated random number sequence.

Definition at line 757 of file random.tcc.

**5.690.4.6** `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 1219 of file random.h.

**5.690.4.7** `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 1206 of file random.h.



**5.690.4.8** `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 1195 of file `random.h`.

## 5.690.5 Friends And Related Function Documentation

**5.690.5.1** `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.

**5.690.5.2** `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 two objects are equal, false otherwise.

Definition at line 1279 of file random.h.

```
5.690.5.3 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](#)

## 5.691 std::slice Class Reference

Class defining one-dimensional subset of an [array](#).

### Public Member Functions

- [slice](#) (size\_t, size\_t, size\_t)
- [slice](#) ()
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

### 5.691.1 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 58 of file [slice\\_array.h](#).

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.692 `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
- 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
- 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
- template<class `_Dom` >  
void **operator**`=` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator**`=` (const `_Tp` &) const
- void **operator**`=` (const `valarray`< `_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**<sup>^</sup>=(const \_Expr<\_Dom, \_Tp > &) const
- void **operator**<sup>^</sup>=(const [valarray](#)<\_Tp > &) const
- `template<class _Dom >`  
void **operator**|=(const \_Expr<\_Dom, \_Tp > &) const
- void **operator**|=(const [valarray](#)<\_Tp > &) const

## Friends

- class [valarray](#)<\_Tp >

### 5.692.1 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 122 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.693 `std::stack< _Tp, _Sequence >` Class Template Reference

A standard container giving FILO 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

- **stack** (`_Sequence &&__c=_Sequence()`)
- **stack** (`const _Sequence &__c`)
- `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 > &`)

### 5.693.1 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 92 of file `stl_stack.h`.

### 5.693.2 Constructor & Destructor Documentation

```
5.693.2.1 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 130 of file `stl_stack.h`.

### 5.693.3 Member Function Documentation

```
5.693.3.1 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 142 of file `stl_stack.h`.

**5.693.3.2** `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 208 of file `stl_stack.h`.

**5.693.3.3** `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 182 of file `stl_stack.h`.

**5.693.3.4** `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 147 of file `stl_stack.h`.

**5.693.3.5** `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 166 of file `stl_stack.h`.

**5.693.3.6** `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 155 of file stl\_stack.h.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 5.694 `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` (const [param\\_type](#) &\_\_p)
- `student_t_distribution` (`_RealType` \_\_n=`_RealType`(1))
- [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::student\\_t\\_distribution](#)< `_RealType1` > &)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & `operator>>` ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &, [std::student\\_t\\_distribution](#)< `_RealType1` > &)

### 5.694.1 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 2718 of file random.h.

### 5.694.2 Member Typedef Documentation

**5.694.2.1** `template<typename _RealType = double> typedef _RealType std::student_t_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2725 of file random.h.

### 5.694.3 Member Function Documentation

**5.694.3.1** `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 2797 of file random.h.

**5.694.3.2** `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 2790 of file random.h.

**5.694.3.3** `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 2783 of file random.h.

**5.694.3.4** `template<typename _RealType = double> param_type  
std::student_t_distribution< _RealType >::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 2775 of file random.h.

**5.694.3.5** `template<typename _RealType = double> void  
std::student_t_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2758 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`, and `std::normal_distribution< _RealType >::reset()`.

**5.694.4 Friends And Related Function Documentation**

**5.694.4.1** `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.

## 5.694 std::student\_t\_distribution<\_RealType> Class Template Reference 3277

**5.694.4.2** `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> &, std::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](#)

## 5.695 `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

#### 5.695.1 Detailed Description

`template<typename _RealType = double> struct std::student_t_distribution<_RealType>::param_type`

Parameter type.

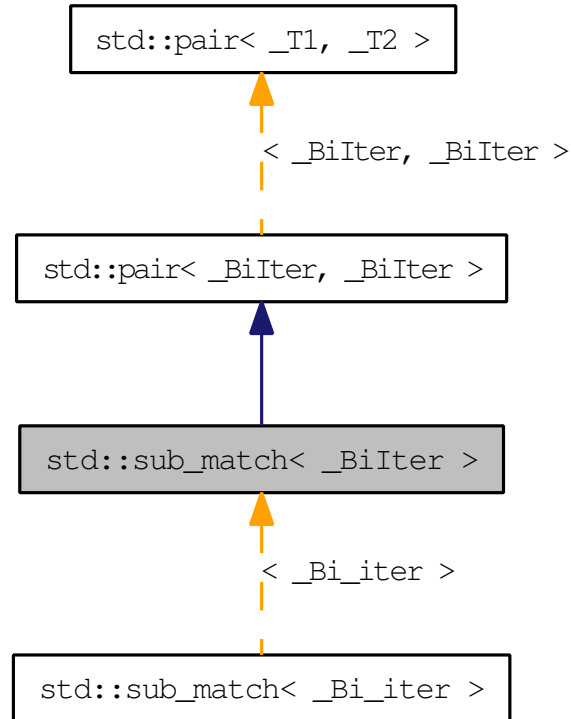
Definition at line 2727 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.696 std::sub\_match< \_BiIter > Class Template Reference

Inheritance diagram for std::sub\_match< \_BiIter >:



### Public Types

- typedef `iterator_traits< _BiIter >::difference_type` **difference\_type**
- typedef `_BiIter` **first\_type**
- typedef `_BiIter` **iterator**
- typedef `_BiIter` **second\_type**
- typedef `iterator_traits< _BiIter >::value_type` **value\_type**

### Public Member Functions

- int `compare` (const `value_type *__s`) const
- int `compare` (const `basic_string< value_type > &__s`) const
- int `compare` (const `sub_match &__s`) const

- difference\_type [length](#) () const
- operator [basic\\_string](#)< value\_type > () const
- [basic\\_string](#)< value\_type > [str](#) () const
- void [swap](#) ([pair](#) &\_\_p)

## Public Attributes

- [\\_BIter](#) [first](#)
- bool [matched](#)
- [\\_BIter](#) [second](#)

### 5.696.1 Detailed Description

`template<typename \_BIter> class std::sub\_match< \_BIter >`

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 1088 of file `tr1_impl/regex`.

### 5.696.2 Member Typedef Documentation

**5.696.2.1** `typedef \_BIter std::pair< \_BIter , \_BIter >::first\_type`  
`[inherited]`

`first_type` is the first bound type

Definition at line 73 of file `stl_pair.h`.

**5.696.2.2** `typedef \_BIter std::pair< \_BIter , \_BIter >::second\_type`  
`[inherited]`

`second_type` is the second bound type

Definition at line 74 of file `stl_pair.h`.



### 5.696.3 Member Function Documentation

**5.696.3.1** `template<typename _BIter> int std::sub_match<_BIter>::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 1172 of file `tr1_impl/regex`.

**5.696.3.2** `template<typename _BIter> int std::sub_match<_BIter>::compare (const basic_string<value_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 1159 of file `tr1_impl/regex`.

**5.696.3.3** `template<typename _BIter> int std::sub_match<_BIter>::compare (const sub_match<_BIter> & __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 1146 of file `tr1_impl/regex`.

Referenced by `std::operator!=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

#### 5.696.3.4 `template<typename _BiIter> difference_type std::sub_match<_BiIter >::length () const [inline]`

Gets the length of the matching sequence.

Definition at line 1103 of file `tr1_impl/regex`.

#### 5.696.3.5 `template<typename _BiIter> std::sub_match<_BiIter >::operator basic_string<value_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 1116 of file `tr1_impl/regex`.

#### 5.696.3.6 `template<typename _BiIter> basic_string<value_type> std::sub_match<_BiIter >::str () const [inline]`

Gets the matching sequence as a string.

**Returns:**

the matching sequence as a string.

Definition at line 1129 of file `tr1_impl/regex`.

Referenced by `std::sub_match<_Bi_iter >::compare()`, `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

## 5.696.4 Member Data Documentation

### 5.696.4.1 `_BiIter std::pair<_BiIter, _BiIter>::first` [inherited]

`first` is a copy of the first object

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

Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator basic_string<value_type>()`, and `std::sub_match<_Bi_iter>::str()`.

### 5.696.4.2 `_BiIter std::pair<_BiIter, _BiIter>::second` [inherited]

`second` is a copy of the second object

Definition at line 77 of file `stl_pair.h`.

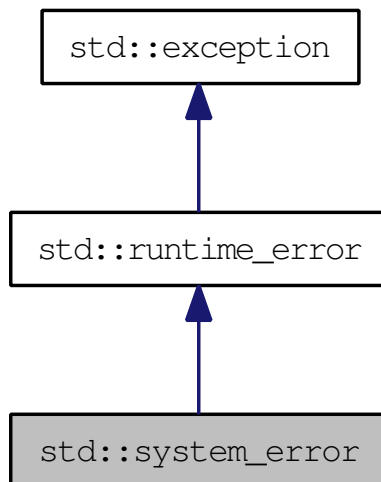
Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator basic_string<value_type>()`, and `std::sub_match<_Bi_iter>::str()`.

The documentation for this class was generated from the following file:

- [tr1\\_impl/regex](#)

## 5.697 `std::system_error` Class Reference

Thrown to indicate error code of underlying system. Inheritance diagram for `std::system_error`:



### Public Member Functions

- `system_error` (int \_\_v, const [error\\_category](#) &\_\_ecat, const [string](#) &\_\_what)
- `system_error` (int \_\_v, const [error\\_category](#) &\_\_ecat)
- `system_error` ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- `system_error` ([error\\_code](#) \_\_ec=[error\\_code](#)())
- const [error\\_code](#) & `code` () const throw ()
- virtual const char \* `what` () const throw ()

### 5.697.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 307 of file `system_error`.

## 5.697.2 Member Function Documentation

### 5.697.2.1 `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](#)

## 5.698 std::thread Class Reference

[thread](#)

### Classes

- class [id](#)  
*thread::id*

### Public Types

- typedef [shared\\_ptr](#)< \_Impl\_base > **\_\_shared\_base\_type**
- typedef [\\_\\_thread\\_t](#) **native\_handle\_type**

### Public Member Functions

- template<typename \_Callable , typename... \_Args>  
**thread** (\_Callable &&\_f, \_Args &&...\_args)
- template<typename \_Callable >  
**thread** (\_Callable \_\_f)
- **thread** ([thread](#) &&\_\_t)
- **thread** (const [thread](#) &)
- void **detach** ()
- [thread::id](#) **get\_id** () const
- void **join** ()
- bool **joinable** () const
- native\_handle\_type **native\_handle** ()
- [thread](#) & **operator=** ([thread](#) &&\_\_t)
- [thread](#) & **operator=** (const [thread](#) &)
- void **swap** ([thread](#) &\_\_t)

### Static Public Member Functions

- static unsigned int **hardware\_concurrency** ()

#### 5.698.1 Detailed Description

[thread](#)

Definition at line 64 of file `thread`.

## 5.698.2 Member Function Documentation

### 5.698.2.1 `native_handle_type std::thread::native_handle()` [`inline`]

**Precondition:**

`thread` is joinable

Definition at line 180 of file `thread`.

The documentation for this class was generated from the following file:

- `thread`

## 5.699 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**

### 5.699.1 Detailed Description

[thread::id](#)

Definition at line 72 of file thread.

The documentation for this class was generated from the following file:

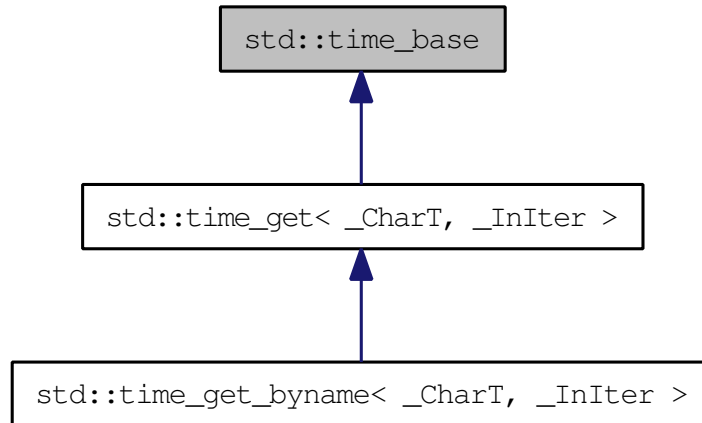
- [thread](#)



## 5.700 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** }

### 5.700.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Definition at line 50 of file `locale_facets_nonio.h`.

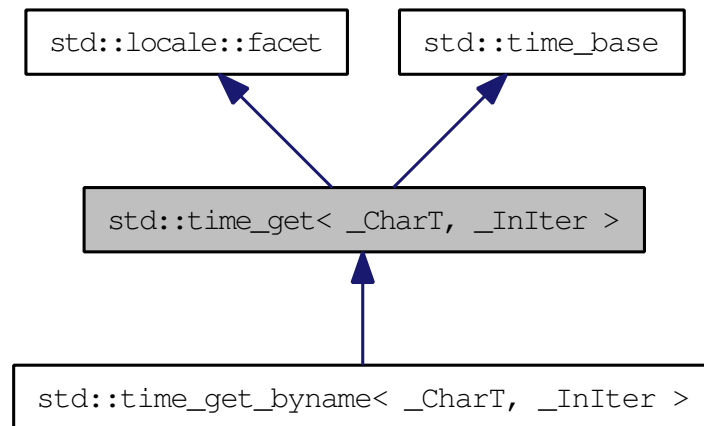
The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.701 `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](#), [ymd](#),  
   [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 ()`
- `__attribute__ ((__const__)) static const char *_S_get_c_name() throw ()`
- `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 &__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 `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale __cloc`, `const char *__s`)

## Friends

- class `locale::_Impl`

### 5.701.1 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 363 of file `locale_facets_nonio.h`.

### 5.701.2 Member Typedef Documentation

**5.701.2.1** `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 369 of file `locale_facets_nonio.h`.

**5.701.2.2** `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 370 of file `locale_facets_nonio.h`.

### 5.701.3 Constructor & Destructor Documentation

**5.701.3.1** `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 385 of file locale\_facets\_nonio.h.

**5.701.3.2** `template<typename _CharT, typename _InIter > virtual std::time_get< _CharT, _InIter >::~time_get () [inline, protected, virtual]`

Destructor.

Definition at line 541 of file locale\_facets\_nonio.h.

### 5.701.4 Member Function Documentation

**5.701.4.1** `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 402 of file locale\_facets\_nonio.h.

**5.701.4.2** `template<typename _CharT, typename _InIter > time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order () const [inline, 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 618 of file `locale_facets_nonio.tcc`.

**5.701.4.3** `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 [inline, 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 1044 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

**5.701.4.4** `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 [inline, 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 1089 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](#).

```
5.701.4.5 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 [inline, 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 1027 of file locale\_facets\_nonio.tcc.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), and [std::ios\\_base::eofbit](#).

**5.701.4.6** `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 [inline, 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 1061 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](#).

**5.701.4.7** `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 [inline, 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 1117 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](#).

```
5.701.4.8 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 451 of file locale\_facets\_nonio.h.

**5.701.4.9** `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 508 of file `locale_facets_nonio.h`.

**5.701.4.10** `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 426 of file locale\_facets\_nonio.h.

```
5.701.4.11 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 479 of file locale\_facets\_nonio.h.

**5.701.4.12** `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 534 of file [locale\\_facets\\_nonio.h](#).

## 5.701.5 Member Data Documentation

**5.701.5.1** `template<typename _CharT , typename _InIter > locale::id  
std::time_get< _CharT, _InIter >::id [inline, static]`

Numpunct facet id.

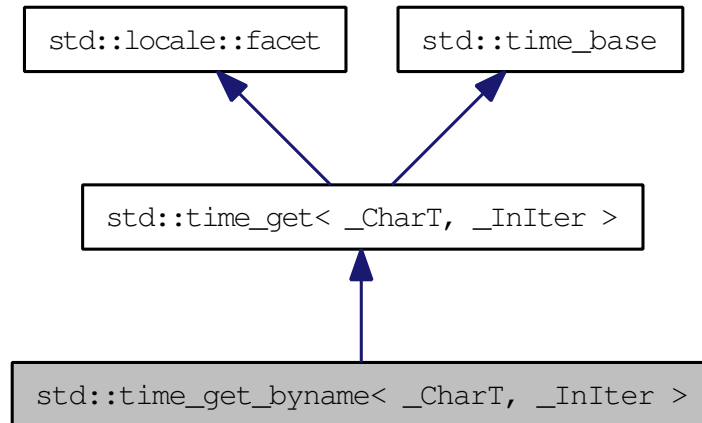
Definition at line 375 of file [locale\\_facets\\_nonio.h](#).

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.702 `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

- `__attribute__((__const__)) static const char *_S_get_c_name() throw ()`
- `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 &__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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.702.1 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 681 of file `locale_facets_nonio.h`.

### 5.702.2 Member Typedef Documentation

**5.702.2.1** `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 685 of file `locale_facets_nonio.h`.

**5.702.2.2** `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 686 of file `locale_facets_nonio.h`.

### 5.702.3 Member Function Documentation

**5.702.3.1** `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 402 of file `locale_facets_nonio.h`.

**5.702.3.2** `template<typename _CharT, typename _InIter >  
time_base::dateorder std::time_get< _CharT, _InIter  
>::do_date_order () const [inline, 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 618 of file `locale_facets_nonio.tcc`.

**5.702.3.3** `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 [inline, 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.



## 5.702 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 3305

Definition at line 1044 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

**5.702.3.4** `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 [inline, protected, virtual,  
inherited]`

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 1089 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`.

**5.702.3.5** `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 [inline, 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.  
*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 1027 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

**5.702.3.6** `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 [inline, 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 1061 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`.

## 5.702 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 3307

**5.702.3.7** `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 [inline, 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 1117 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](#).

**5.702.3.8** `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 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 451 of file locale\_facets\_nonio.h.

**5.702.3.9** `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 508 of file locale\_facets\_nonio.h.

## 5.702 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 3309

**5.702.3.10** `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 426 of file `locale_facets_nonio.h`.

**5.702.3.11** `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 479 of file locale\_facets\_nonio.h.

**5.702.3.12** `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 534 of file locale\_facets\_nonio.h.

**5.702.4 Member Data Documentation**

**5.702.4.1** `template<typename _CharT, typename _InIter > locale::id  
std::time_get< _CharT, _InIter >::id [inline, static,  
inherited]`

Numpunct facet id.

## **5.702 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 3311**

Definition at line 375 of file locale\_facets\_nonio.h.

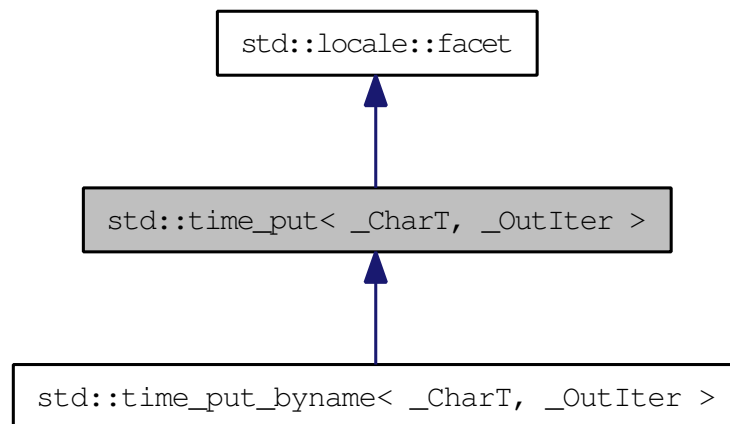
The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.703 `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](#) ()
- **\_\_attribute\_\_** ((\_\_const\_\_)) static const char \*\_S\_get\_c\_name() throw ()
- 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 \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Friends

- class [locale::\\_Impl](#)

### 5.703.1 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 710 of file `locale_facets_nonio.h`.

### 5.703.2 Member Typedef Documentation

**5.703.2.1** **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 716 of file `locale_facets_nonio.h`.

### 5.703.2.2 `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 717 of file locale\_facets\_nonio.h.

## 5.703.3 Constructor & Destructor Documentation

### 5.703.3.1 `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 731 of file locale\_facets\_nonio.h.

### 5.703.3.2 `template<typename _CharT, typename _OutIter > virtual std::time_put< _CharT, _OutIter >::~~time_put () [inline, protected, virtual]`

Destructor.

Definition at line 777 of file locale\_facets\_nonio.h.

## 5.703.4 Member Function Documentation

### 5.703.4.1 `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 [inline, 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 1175 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\(\)](#).

```
5.703.4.2 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 770 of file locale\_facets\_nonio.h.

References [std::time\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#).

**5.703.4.3** `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 [inline]`

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 1140 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\(\)](#).

## 5.703.5 Member Data Documentation

**5.703.5.1** `template<typename _CharT, typename _OutIter > locale::id  
std::time_put< _CharT, _OutIter >::id [inline, static]`

Numpunct facet id.

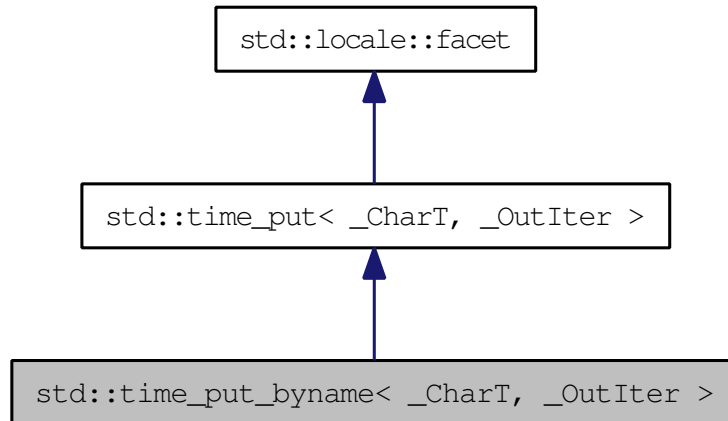
Definition at line 721 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.704 `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`
- `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

- `__attribute__` (`((__const__)) static const char * _S_get_c_name()` `throw ()`)

- 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 `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

## 5.704.1 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 806 of file `locale_facets_nonio.h`.

## 5.704.2 Member Typedef Documentation

**5.704.2.1** `template<typename _CharT, typename _OutIter > typedef _CharT std::time_put_byname<_CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented from [std::time\\_put<\\_CharT, \\_OutIter >](#).

Definition at line 810 of file `locale_facets_nonio.h`.

**5.704.2.2** `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 811 of file `locale_facets_nonio.h`.

### 5.704.3 Member Function Documentation

**5.704.3.1** `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 [inline, 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.

Definition at line 1175 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()`.

**5.704.3.2** `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 770 of file `locale_facets_nonio.h`.

References `std::time_put<_CharT, _OutIter >::do_put()`.

**5.704.3.3** `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 [inline, 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 1140 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()`.



## 5.704 std::time\_put\_byname< \_CharT, \_OutIter > Class Template Reference 3321

### 5.704.4 Member Data Documentation

**5.704.4.1** `template<typename _CharT , typename _OutIter > locale::id  
std::time_put< _CharT, _OutIter >::id [inline, static,  
inherited]`

Numpunct facet id.

Definition at line 721 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.705 `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` ()

### 5.705.1 Detailed Description

[timed\\_mutex](#)

Definition at line 167 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.706 `std::try_to_lock_t` Struct Reference

Try to acquire ownership of the [mutex](#) without blocking.

### 5.706.1 Detailed Description

Try to acquire ownership of the [mutex](#) without blocking.

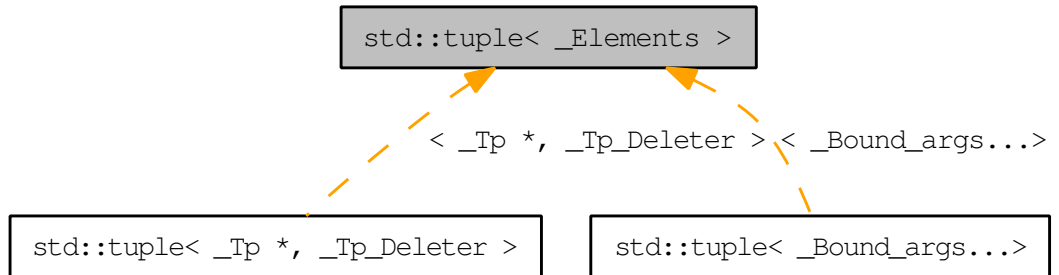
Definition at line 379 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.707 `std::tuple< _Elements >` Class Template Reference

[tuple](#) Inheritance diagram for `std::tuple< _Elements >`:



### Public Member Functions

- `template<typename... _UElements>`  
**tuple** ([tuple](#)< \_UElements...> &\_\_in)
- `template<typename... _UElements>`  
**tuple** ([tuple](#)< \_UElements...> &&\_\_in)
- `template<typename... _UElements>`  
**tuple** (const [tuple](#)< \_UElements...> &\_\_in)
- **tuple** ([tuple](#) &&\_\_in)
- **tuple** (const [tuple](#) &\_\_in)
- `template<typename... _UElements>`  
**tuple** (\_UElements &&...\_\_elements)
- **tuple** (const \_Elements &&...\_\_elements)
- `template<typename... _UElements>`  
[tuple](#) & **operator=** ([tuple](#)< \_UElements...> &&\_\_in)
- `template<typename... _UElements>`  
[tuple](#) & **operator=** (const [tuple](#)< \_UElements...> &\_\_in)
- [tuple](#) & **operator=** ([tuple](#) &&\_\_in)
- [tuple](#) & **operator=** (const [tuple](#) &\_\_in)
- void **swap** ([tuple](#) &\_\_in)

### 5.707.1 Detailed Description

`template<typename... _Elements> class std::tuple< _Elements >`

[tuple](#)

Definition at line 224 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.708 `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

- `template<typename _U1, typename _U2 >`  
`tuple` (`pair< _U1, _U2 > &&__in`)
- `template<typename _U1, typename _U2 >`  
`tuple` (`const pair< _U1, _U2 > &__in`)
- `template<typename _U1, typename _U2 >`  
`tuple` (`tuple< _U1, _U2 > &&__in`)
- `template<typename _U1, typename _U2 >`  
`tuple` (`const tuple< _U1, _U2 > &__in`)
- `tuple` (`tuple &&__in`)
- `tuple` (`const tuple &__in`)
- `template<typename _U1, typename _U2 >`  
`tuple` (`_U1 &&__a1, _U2 &&__a2`)
- `tuple` (`const _T1 &__a1, const _T2 &__a2`)
- `template<typename _U1, typename _U2 >`  
`tuple & operator=` (`pair< _U1, _U2 > &&__in`)
- `template<typename _U1, typename _U2 >`  
`tuple & operator=` (`const pair< _U1, _U2 > &__in`)
- `template<typename _U1, typename _U2 >`  
`tuple & operator=` (`tuple< _U1, _U2 > &&__in`)
- `template<typename _U1, typename _U2 >`  
`tuple & operator=` (`const tuple< _U1, _U2 > &__in`)
- `tuple & operator=` (`tuple &&__in`)
- `tuple & operator=` (`const tuple &__in`)
- `void swap` (`tuple &__in`)

### 5.708.1 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 307 of file `tuple`.

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.709 `std::tuple_element< 0, tuple< _Head, _Tail...>` > Struct Template Reference

### Public Types

- `typedef _Head type`

### 5.709.1 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 420 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)



## 5.710 `std::tuple_element< __i, tuple< _Head, _Tail...> >` Struct Template Reference

Inherits `tuple_element< __i-1, tuple< _Tail...> >`.

### 5.710.1 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 413 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.711 `std::tuple_size< tuple< _Elements...> >` Struct Template Reference

class `tuple_size`

### Static Public Attributes

- static const `std::size_t` value

### 5.711.1 Detailed Description

`template<typename... _Elements> struct std::tuple_size< tuple< _Elements...> >`

class `tuple_size`

Definition at line 431 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.712 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`
- `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`

#### 5.712.1 Detailed Description

Part of RTTI. The `type_info` class describes type information generated by an implementation.

Definition at line 87 of file `typeinfo`.

#### 5.712.2 Constructor & Destructor Documentation

##### 5.712.2.1 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.

### 5.712.3 Member Function Documentation

#### 5.712.3.1 `const char* std::type_info::name () const` [`inline`]

Returns an *implementation-defined* byte string; this is not portable between compilers!

Definition at line 98 of file `typeinfo`.

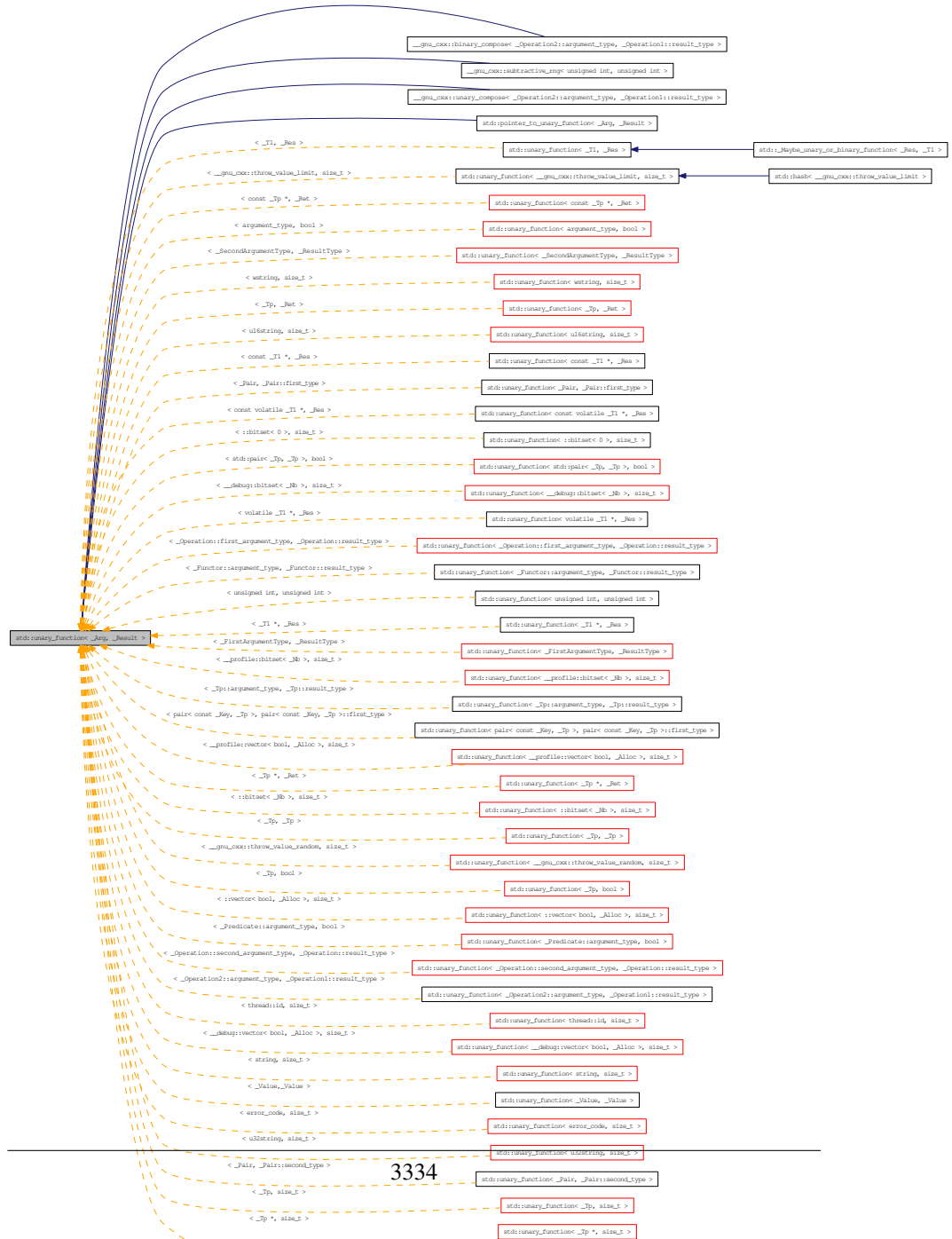
The documentation for this class was generated from the following file:

- [typeinfo](#)



## 5.713 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](#)

### 5.713.1 Detailed Description

`template<typename _Arg, typename _Result> struct std::unary_function< _Arg, _Result >`

This is one of the [functor base classes](#).

Definition at line 100 of file `stl_function.h`.

### 5.713.2 Member Typedef Documentation

**5.713.2.1** `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.713.2.2** `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`

`result_type` is the return type

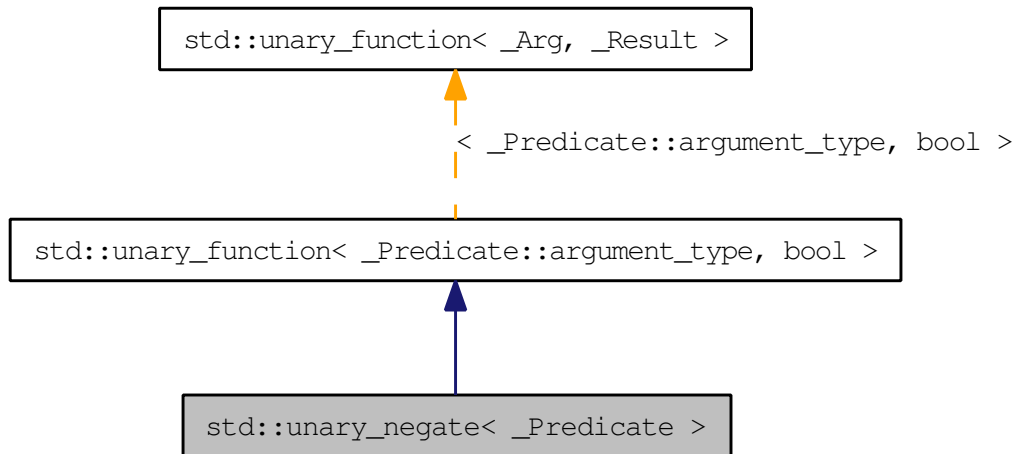
Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.714 `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`

#### 5.714.1 Detailed Description

```
template<typename _Predicate> class std::unary_negate< _Predicate >
```

One of the [negation functors](#).



Definition at line 346 of file `stl_function.h`.

## 5.714.2 Member Typedef Documentation

### 5.714.2.1 `typedef _Predicate::argument_type std::unary_function<_Predicate::argument_type, bool >::argument_type` [`inherited`]

`argument_type` is the type of the argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.714.2.2 `typedef bool std::unary_function<_Predicate::argument_type, bool >::result_type` [`inherited`]

`result_type` is the return type

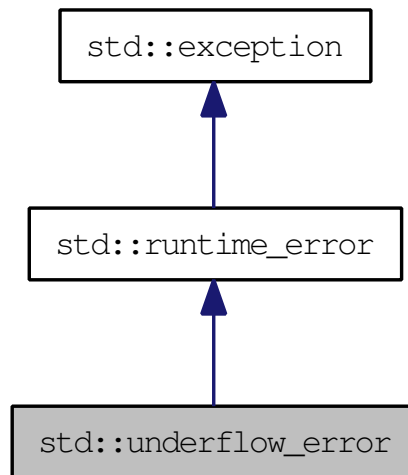
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.715 `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 ()

#### 5.715.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 140 of file `stdexcept`.

#### 5.715.2 Member Function Documentation

##### 5.715.2.1 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](#)

## 5.716 `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** (const [param\\_type](#) &\_\_p)
- **uniform\_int\_distribution** (`_IntType` \_\_a=0, `_IntType` \_\_b=`std::numeric_limits<_IntType >::max()`)
- **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**

## 5.716 std::uniform\_int\_distribution< \_IntType > Class Template Reference 3341

### 5.716.1 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 1496 of file random.h.

### 5.716.2 Member Typedef Documentation

**5.716.2.1** `template<typename _IntType = int> typedef _IntType  
std::uniform_int_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 1503 of file random.h.

### 5.716.3 Constructor & Destructor Documentation

**5.716.3.1** `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 1535 of file random.h.

### 5.716.4 Member Function Documentation

**5.716.4.1** `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 1572 of file random.h.

**5.716.4.2** `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 1565 of file random.h.

**5.716.4.3** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > uniform_int_distribution<  
_IntType >::result_type std::uniform_int_distribution< _IntType  
>::operator() (_UniformRandomNumberGenerator & __urng, const  
param_type & __p) [inline]`

Gets a uniform random number in the range  $[0, n)$ .

This function is aimed at use with `std::random_shuffle`.

Definition at line 818 of file random.tcc.

**5.716.4.4** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::uniform_int_distribution< _IntType >::operator()  
(_UniformRandomNumberGenerator & __urng) [inline]`

Gets a uniformly distributed random number in the range  $(min, max)$ .

Definition at line 1596 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()`.

**5.716.4.5** `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 1587 of file random.h.

## 5.716 std::uniform\_int\_distribution< \_IntType > Class Template Reference 3343

**5.716.4.6** `template<typename _IntType = int> param_type  
std::uniform_int_distribution< _IntType >::param () const  
[inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 1579 of file random.h.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`, and `std::operator>>()`.

**5.716.4.7** `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 1551 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.717 `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

### 5.717.1 Detailed Description

`template<typename _IntType = int> struct std::uniform_int_distribution< _IntType >::param_type`

Parameter type.

Definition at line 1505 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)



## 5.718 `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** (const [param\\_type](#) &\_\_p)
- **uniform\_real\_distribution** (`_RealType` \_\_a=`_RealType`(0), `_RealType` \_\_b=`_RealType`(1))
- [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** ()

### 5.718.1 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).

Definition at line 1650 of file random.h.

## 5.718.2 Member Typedef Documentation

**5.718.2.1** `template<typename _RealType = double> typedef _RealType  
std::uniform_real_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 1657 of file random.h.

## 5.718.3 Constructor & Destructor Documentation

**5.718.3.1** `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 1692 of file random.h.

## 5.718.4 Member Function Documentation

**5.718.4.1** `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 1729 of file random.h.

**5.718.4.2** `template<typename _RealType = double> result_type  
std::uniform_real_distribution< _RealType >::min () const  
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1722 of file random.h.

## 5.718 std::uniform\_real\_distribution<\_RealType> Class Template Reference

**5.718.4.3** `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 1744 of file `random.h`.

**5.718.4.4** `template<typename _RealType = double> param_type  
std::uniform_real_distribution<_RealType>::param () const  
[inline]`

Returns the parameter `set` of the distribution.

Definition at line 1736 of file `random.h`.

Referenced by `std::operator>>()`.

**5.718.4.5** `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 1708 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.719 `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

### 5.719.1 Detailed Description

```
template<typename _RealType = double> struct std::uniform_real_
distribution< _RealType >::param_type
```

Parameter type.

Definition at line 1659 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.720 `std::unique_lock< _Mutex >` Class Template Reference

[unique\\_lock](#)

### Public Types

- typedef `_Mutex` `mutex_type`

### Public Member Functions

- `unique_lock` (`unique_lock` &&\_\_u)
- `unique_lock` (const `unique_lock` &)
- template<typename `_Rep`, typename `_Period` >  
`unique_lock` (`mutex_type` &\_\_m, const `chrono::duration`< `_Rep`, `_Period` > &\_\_rtime)
- template<typename `_Clock`, typename `_Duration` >  
`unique_lock` (`mutex_type` &\_\_m, const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_atime)
- `unique_lock` (`mutex_type` &\_\_m, `adopt_lock_t`)
- `unique_lock` (`mutex_type` &\_\_m, `try_to_lock_t`)
- `unique_lock` (`mutex_type` &\_\_m, `defer_lock_t`)
- `unique_lock` (`mutex_type` &\_\_m)
- void `lock` ()
- `mutex_type` \* `mutex` () const
- `operator bool` () const
- `unique_lock` & `operator=` (`unique_lock` &&\_\_u)
- `unique_lock` & `operator=` (const `unique_lock` &)
- 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` ()

### 5.720.1 Detailed Description

**template<typename \_Mutex> class std::unique\_lock< \_Mutex >**

[unique\\_lock](#)

Definition at line 416 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.721 `std::unique_ptr< _Tp, _Tp_Deleter >` Class Template Reference

20.7.12.2 [unique\\_ptr](#) for single objects.

### Public Types

- typedef `_Tp_Deleter` **deleter\_type**
- typedef `_Tp` **element\_type**
- typedef `_Tp *` **pointer**

### Public Member Functions

- **unique\_ptr** (const [unique\\_ptr](#) &)
- template<typename `_Up`, typename `_Up_Deleter` >  
**unique\_ptr** ([unique\\_ptr](#)< `_Up`, `_Up_Deleter` > &&\_\_u)
- **unique\_ptr** ([unique\\_ptr](#) &&\_\_u)
- **unique\_ptr** (pointer `__p`, typename [std::remove\\_reference](#)< `deleter_type` >::type &&\_\_d)
- **unique\_ptr** (pointer `__p`, typename [std::conditional](#)< [std::is\\_reference](#)< `deleter_type` >::value, `deleter_type`, const `deleter_type` & >::type `__d`)
- **unique\_ptr** (pointer `__p`)
- 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=** (`__unspecified_pointer_type`)
- template<typename `_Up`, typename `_Up_Deleter` >  
[unique\\_ptr](#) & **operator=** ([unique\\_ptr](#)< `_Up`, `_Up_Deleter` > &&\_\_u)
- [unique\\_ptr](#) & **operator=** ([unique\\_ptr](#) &&\_\_u)
- pointer **release** ()
- void **reset** (pointer `__p`=pointer())
- void **swap** ([unique\\_ptr](#) &\_\_u)

### 5.721.1 Detailed Description

```
template<typename _Tp, typename _Tp_Deleter = default_delete<_Tp>> class
std::unique_ptr<_Tp, _Tp_Deleter >
```

20.7.12.2 [unique\\_ptr](#) for single objects.

Definition at line 81 of file [unique\\_ptr.h](#).

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)



## 5.722 `std::unique_ptr< _Tp[], _Tp_Deleter >` Class Template Reference

20.7.12.3 `unique_ptr` for `array` objects with a runtime length

### Public Types

- typedef `_Tp_Deleter` **deleter\_type**
- typedef `_Tp` **element\_type**
- typedef `_Tp *` **pointer**

### Public Member Functions

- `template<typename _Up >`  
**unique\_ptr** (`_Up *`, `typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type *=0`)
- `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`)
- `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`)
- **unique\_ptr** (`const unique_ptr &`)
- `template<typename _Up, typename _Up_Deleter >`  
**unique\_ptr** (`unique_ptr< _Up, _Up_Deleter > &&__u`)
- **unique\_ptr** (`unique_ptr &&__u`)
- **unique\_ptr** (`pointer __p`, `typename std::remove_reference< deleter_type >::type &&__d`)
- **unique\_ptr** (`pointer __p`, `typename std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- **unique\_ptr** (`pointer __p`)
- `pointer` **get** () const
- `const deleter_type &` **get\_deleter** () const
- `deleter_type &` **get\_deleter** ()
- **operator bool** () const
- `unique_ptr &` **operator=** (`const unique_ptr &`)
- `unique_ptr &` **operator=** (`__unspecified_pointer_type`)
- `template<typename _Up, typename _Up_Deleter >`  
`unique_ptr &` **operator=** (`unique_ptr< _Up, _Up_Deleter > &&__u`)
- `unique_ptr &` **operator=** (`unique_ptr &&__u`)

- [std::add\\_lvalue\\_reference](#)< element\_type >::type **operator**[ ] (size\_t \_\_i) const
- pointer **release** ()
- `template<typename _Up >`  
void **reset** (\_Up)
- void **reset** (pointer \_\_p=pointer())
- void **swap** ([unique\\_ptr](#) &\_\_u)

### 5.722.1 Detailed Description

`template<typename _Tp, typename _Tp_Deleter> class std::unique_ptr< _Tp[ ],  
_Tp_Deleter >`

20.7.12.3 [unique\\_ptr](#) for [array](#) objects with a runtime length

Definition at line 219 of file [unique\\_ptr.h](#).

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 5.723 `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 _Base::hasher hasher`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- `unordered_map` ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_map` ([unordered\\_map](#) &&\_\_x)
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_map` (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_map` & `operator=` ([initializer\\_list](#)< value\_type > \_\_l)
- `unordered_map` & `operator=` ([unordered\\_map](#) &&\_\_x)

### 5.723.1 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 185 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.724 `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 _Base::hasher hasher`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- `unordered_multimap` ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multimap` ([unordered\\_multimap](#) &&\_\_x)
- `template<typename _InputIterator >`  
`unordered_multimap` (\_InputIterator \_\_f, \_InputIterator \_\_l, typename \_Base::size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multimap` (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multimap` & `operator=` ([initializer\\_list](#)< value\_type > \_\_l)
- `unordered_multimap` & `operator=` ([unordered\\_multimap](#) &&\_\_x)

### 5.724.1 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 266 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.725 `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 _Base::hasher hasher`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- `unordered_multiset` ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multiset` ([unordered\\_multiset](#) &&\_\_x)
- `template<typename _InputIterator >`  
`unordered_multiset` (\_InputIterator \_\_f, \_InputIterator \_\_l, typename \_Base::size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multiset` (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `unordered_multiset` & `operator=` ([initializer\\_list](#)< value\_type > \_\_l)
- `unordered_multiset` & `operator=` ([unordered\\_multiset](#) &&\_\_x)

### 5.725.1 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 256 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)



## 5.726 `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 _Base::hasher hasher`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- `unordered_set` (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_set` (`unordered_set &&__x`)
- `template<typename _InputIterator >`  
`unordered_set` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_set` (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_set & operator=` (`initializer_list< value_type > __l`)
- `unordered_set & operator=` (`unordered_set &&__x`)

### 5.726.1 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>.

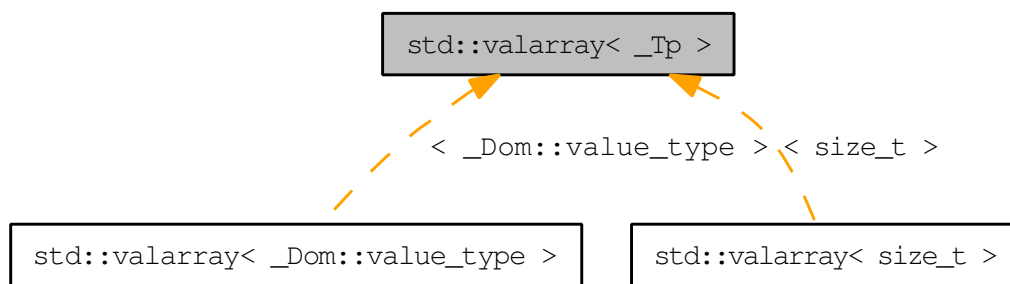
Definition at line 178 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.727 `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

- `template<typename _Tp>`  
**valarray** (const `_Tp` \*\_\_restrict\_\_, size\_t \_\_n)
- `template<class _Dom >`  
**valarray** (const `_Expr<_Dom, _Tp>` &\_\_e)
- **valarray** (`initializer_list<_Tp>`)
- **valarray** (const `indirect_array<_Tp>` &)
- **valarray** (const `mask_array<_Tp>` &)
- **valarray** (const `gslice_array<_Tp>` &)
- **valarray** (const `slice_array<_Tp>` &)
- **valarray** (const `valarray` &)
- **valarray** (const `_Tp` \*\_\_restrict\_\_, size\_t)
- **valarray** (const `_Tp` &, size\_t)
- **valarray** (size\_t)
- **valarray** ()
- `_Expr<_RefFunClos<_ValArray, _Tp>, _Tp>` **apply** (`_Tp` func(const `_Tp` &)) const
- `_Expr<_ValFunClos<_ValArray, _Tp>, _Tp>` **apply** (`_Tp` func(`_Tp`)) const
- `valarray<_Tp>` **cshift** (int) const
- `_Tp` **max** () const
- `_Tp` **min** () const

- `_UnaryOp< __logical_not >::_Rt operator! () const`
- `template<class _Dom >`  
`valarray< _Tp > & operator%= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator%= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator%= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator&= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator&= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator&= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator*= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator*= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator*= (const _Tp &)`
- `_UnaryOp< __unary_plus >::_Rt operator+ () const`
- `template<class _Dom >`  
`valarray< _Tp > & operator+= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator+= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator+= (const _Tp &)`
- `_UnaryOp< __negate >::_Rt operator- () const`
- `template<class _Dom >`  
`valarray< _Tp > & operator-= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator-= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator-= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator/= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator/= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator/= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator<<= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator<<= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator<<= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator= (const _Expr< _Dom, _Tp > &)`
- `valarray & operator= (initializer_list< _Tp >)`
- `valarray< _Tp > & operator= (const indirect_array< _Tp > &)`
- `valarray< _Tp > & operator= (const mask_array< _Tp > &)`
- `valarray< _Tp > & operator= (const gslice_array< _Tp > &)`
- `valarray< _Tp > & operator= (const slice_array< _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 valarray< _Tp > &)`

- `valarray<_Tp> & operator>>=` (const `_Tp` &)
- `indirect_array<_Tp> operator[]` (const `valarray<size_t>` &)
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> operator[]` (const `valarray<size_t>` &) const
- `mask_array<_Tp> operator[]` (const `valarray<bool>` &)
- `valarray<_Tp> operator[]` (const `valarray<bool>` &) const
- `gslice_array<_Tp> operator[]` (const `gslice` &)
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> operator[]` (const `gslice` &) const
- `slice_array<_Tp> operator[]` (`slice`)
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> operator[]` (`slice`) const
- `const _Tp & operator[]` (`size_t`) const
- `_Tp & operator[]` (`size_t`)
- `template<class _Dom>`  
`valarray<_Tp> & operator^=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator^=` (const `valarray<_Tp>` &)
- `valarray<_Tp> & operator^=` (const `_Tp` &)
- `template<class _Dom>`  
`valarray<_Tp> & operator|=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator|=` (const `valarray<_Tp>` &)
- `valarray<_Tp> & operator|=` (const `_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>`

### 5.727.1 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 112 of file `valarray`.

## 5.727.2 Constructor & Destructor Documentation

### 5.727.2.1 `template<class _Tp> std::valarray<_Tp>::valarray (const _Tp * __restrict__, size_t)`

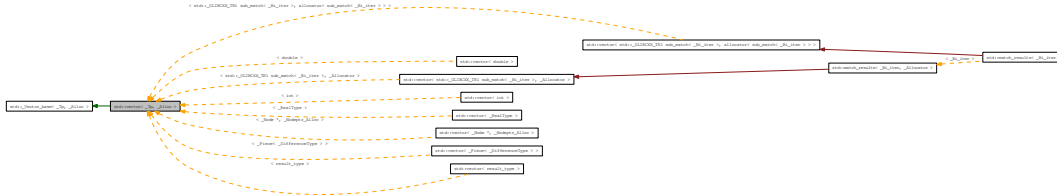
Construct an [array](#) initialized to the first  $n$  elements of  $t$ .

The documentation for this class was generated from the following file:

- [valarray](#)

## 5.728 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

- template<typename \_InputIterator >  
**vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- **vector** (**initializer\_list**< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **vector** (**vector** &&\_\_x)
- **vector** (const **vector** &\_\_x)
- **vector** (size\_type \_\_n, const value\_type &\_\_value=value\_type(), const allocator\_type &\_\_a=allocator\_type())

- [vector](#) (const allocator\_type &\_\_a)
- [vector](#) ()
- [~vector](#) ()
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- const\_reference [back](#) () const
- reference [back](#) ()
- 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
- const\_pointer [data](#) () const
- pointer [data](#) ()
- template<typename... \_Args>  
iterator [emplace](#) (iterator \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- const\_iterator [end](#) () const
- iterator [end](#) ()
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- iterator [erase](#) (iterator \_\_position)
- const\_reference [front](#) () const
- reference [front](#) ()
- template<typename \_InputIterator >  
void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- void [insert](#) (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- iterator [insert](#) (iterator \_\_position, 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)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const



- reference `operator[]` (size\_type \_\_n)
- void `pop_back` ()
- void `push_back` (value\_type &&\_\_x)
- void `push_back` (const value\_type &\_\_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 \_\_new\_size, value\_type \_\_x=value\_type())
- 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 _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 _InputIterator >`  
void `_M_assign_dispatch` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_false_type`)
- `template<typename _Integer >`  
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_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` ()
- `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`

### 5.728.1 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 170 of file `stl_vector.h`.

## 5.728.2 Constructor & Destructor Documentation

**5.728.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector () [inline]`

Default constructor creates no elements.

Definition at line 207 of file stl\_vector.h.

**5.728.2.2** `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 215 of file stl\_vector.h.

**5.728.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector (size_type __n, const value_type  
& __value = value_type (), const allocator_type & __a =  
allocator_type ()) [inline, explicit]`

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 227 of file stl\_vector.h.

**5.728.2.4** `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 241 of file `stl_vector.h`.

```
5.728.2.5 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 257 of file `stl_vector.h`.

```
5.728.2.6 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 271 of file `stl_vector.h`.

```
5.728.2.7 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 297 of file `stl_vector.h`.

**5.728.2.8** `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 312 of file `stl_vector.h`.

### 5.728.3 Member Function Documentation

**5.728.3.1** `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 964 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc >::operator=()`.

**5.728.3.2** `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 639 of file `stl_vector.h`.

```
5.728.3.3 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 412 of file `stl_vector.h`.

Referenced by `std::vector< result_type >::assign()`.

```
5.728.3.4 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 392 of file `stl_vector.h`.

```
5.728.3.5 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 375 of file `stl_vector.h`.

```
5.728.3.6 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 676 of file `stl_vector.h`.

```
5.728.3.7 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 658 of file `stl_vector.h`.

**5.728.3.8** `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 711 of file `stl_vector.h`.

**5.728.3.9** `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 703 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType >::max()`, and `std::piecewise_constant_distribution<_RealType >::max()`.

**5.728.3.10** `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 >`, and `std::match_results<_Bi_iter >`.

Definition at line 435 of file `stl_vector.h`.

**5.728.3.11** `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 426 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc >::emplace()`, `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==()`, and `std::vector<result_type >::vector()`.



**5.728.3.12** `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 573 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc>::operator=()`.

**5.728.3.13** `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>](#), and [std::match\\_results<\\_Bi\\_iter>](#).

Definition at line 499 of file `stl_vector.h`.

**5.728.3.14** `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>](#), and [std::match\\_results<\\_Bi\\_iter>](#).

Definition at line 508 of file `stl_vector.h`.

**5.728.3.15** `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 954 of file `stl_vector.h`.

**5.728.3.16** `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 517 of file `stl_vector.h`.

**5.728.3.17** `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 526 of file `stl_vector.h`.

**5.728.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
pointer 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 722 of file `stl_vector.h`.

**5.728.3.19** `template<typename _Tp, typename _Alloc> template<typename...  
_Args> vector<_Tp, _Alloc>::iterator vector::emplace(iterator  
_position, _Args&&... _args) [inline]`

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 272 of file `vector.tcc`.

References `std::vector<_Tp, _Alloc >::begin()`, and `std::vector<_Tp, _Alloc >::end()`.

**5.728.3.20** `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 >`, and `std::match_results<_Bi_iter >`.

Definition at line 582 of file `stl_vector.h`.

**5.728.3.21** `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 >`, and `std::match_results<_Bi_iter >`.

Definition at line 453 of file `stl_vector.h`.

**5.728.3.22** `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 444 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc >::emplace()`, `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==()`, and `std::vector<result_type >::vector()`.

**5.728.3.23** `template<typename _Tp, typename _Alloc > vector<_Tp,  
_Alloc >::iterator vector::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 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 146 of file vector.tcc.

References [std::vector<\\_Tp, \\_Alloc >::end\(\)](#).

**5.728.3.24** `template<typename _Tp, typename _Alloc > vector<_Tp, _Alloc >::iterator vector::erase (iterator __position) [inline]`

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 134 of file vector.tcc.

References [std::vector<\\_Tp, \\_Alloc >::end\(\)](#).

**5.728.3.25** `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 695 of file stl\_vector.h.

**5.728.3.26** `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 687 of file stl\_vector.h.

Referenced by `std::piecewise_linear_distribution< _RealType >::min()`, and `std::piecewise_constant_distribution< _RealType >::min()`.

**5.728.3.27** `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 877 of file stl\_vector.h.

**5.728.3.28** `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 858 of file `stl_vector.h`.

```
5.728.3.29 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 840 of file `stl_vector.h`.

Referenced by `std::vector< result_type >::insert()`.

```
5.728.3.30 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 823 of file `stl_vector.h`.

**5.728.3.31** `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator vector::insert (iterator __position, const value_type & __x) [inline]`

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 107 of file vector.tcc.

References [std::vector< \\_Tp, \\_Alloc >::begin\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::end\(\)](#).

**5.728.3.32** `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.

Definition at line 538 of file stl\_vector.h.

**5.728.3.33** `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 357 of file stl\_vector.h.

**5.728.3.34** `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 336 of file `stl_vector.h`.

**5.728.3.35** `template<typename _Tp, typename _Alloc > vector<_Tp, _Alloc  
> & vector::operator= (const vector<_Tp, _Alloc > & _x)  
[inline]`

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 157 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()`.

**5.728.3.36** `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\(\)](#).)

Reimplemented in [std::match\\_results<\\_Bi\\_iter>](#).

Definition at line 633 of file `stl_vector.h`.

**5.728.3.37** `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 618 of file `stl_vector.h`.

**5.728.3.38** `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 772 of file `stl_vector.h`.

**5.728.3.39** `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 741 of file `stl_vector.h`.

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

**5.728.3.40** `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 471 of file `stl_vector.h`.

**5.728.3.41** `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 462 of file `stl_vector.h`.

**5.728.3.42** `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 489 of file `stl_vector.h`.

**5.728.3.43** `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 480 of file `stl_vector.h`.

**5.728.3.44** `template<typename _Tp, typename _Alloc> void vector::reserve`  
`(size_type __n) [inline]`

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 65 of file `vector.tcc`.

References `std::_Destroy()`.

**5.728.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`void std::vector<_Tp, _Alloc>::resize (size_type __new_size,`  
`value_type __x = value_type()) [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 553 of file `stl_vector.h`.

Referenced by `__gnu_parallel::_shrink_and_double()`, `__gnu_parallel::multiway_merge_exact_splitting()`, and `__gnu_parallel::parallel_sort_mwms()`.

**5.728.3.46** `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 564 of file `stl_vector.h`.

**5.728.3.47** `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>`, and `std::match_results<_Bi_iter>`.

Definition at line 533 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution<_IntType>::max()`, `std::vector<_Tp, _Alloc>::operator=()`, and `std::operator==(())`.

**5.728.3.48** `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 934 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 5.729 `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

- `template<typename _InputIterator >`  
**vector** (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a=allocator_type()`)
- **vector** (`initializer_list< bool > __l`, `const allocator_type &__a=allocator_type()`)
- **vector** (`vector &&__x`)
- **vector** (`const vector &__x`)
- **vector** (`size_type __n`, `const bool &__value=bool()`, `const allocator_type &__a=allocator_type()`)
- **vector** (`const allocator_type &__a`)
- void **assign** (`initializer_list< bool > __l`)
- `template<typename _InputIterator >`  
void **assign** (`_InputIterator __first`, `_InputIterator __last`)
- void **assign** (`size_type __n`, `const bool &__x`)
- `const_reference` **at** (`size_type __n`) `const`
- `reference` **at** (`size_type __n`)
- `const_reference` **back** () `const`

- reference **back** ()
- const\_iterator **begin** () const
- iterator **begin** ()
- size\_type **capacity** () const
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- void **clear** ()
- [const\\_reverse\\_iterator](#) **crbegin** () const
- [const\\_reverse\\_iterator](#) **crend** () const
- void **data** ()
- bool **empty** () const
- const\_iterator **end** () const
- iterator **end** ()
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- void **flip** ()
- const\_reference **front** () const
- reference **front** ()
- allocator\_type **get\_allocator** () const
- void **insert** (iterator \_\_p, [initializer\\_list](#)< bool > \_\_l)
- void **insert** (iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (iterator \_\_position, const bool &\_\_x=bool())
- size\_type **max\_size** () const
- [vector](#) & **operator=** ([initializer\\_list](#)< bool > \_\_l)
- [vector](#) & **operator=** ([vector](#) &&\_\_x)
- [vector](#) & **operator=** (const [vector](#) &\_\_x)
- const\_reference **operator[]** (size\_type \_\_n) const
- reference **operator[]** (size\_type \_\_n)
- void **pop\_back** ()
- void **push\_back** (bool \_\_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 \_\_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 `_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<class `_InputIterator` >  
void **\_M\_assign\_dispatch** (`_InputIterator` `__first`, `_InputIterator` `__last`, `__false_type`)
- template<typename `_Integer` >  
void **\_M\_assign\_dispatch** (`_Integer` `__n`, `_Integer` `__val`, `__true_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`)
- const `_Bit_alloc_type` & **\_M\_get\_Bit\_allocator** () const
- `_Bit_alloc_type` & **\_M\_get\_Bit\_allocator** ()
- 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 `_ForwardIterator` >  
void **\_M\_initialize\_range** (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, [std::forward\\_iterator\\_tag](#))
- template<typename `_InputIterator` >  
void **\_M\_initialize\_range** (`_InputIterator` `__first`, `_InputIterator` `__last`, [std::input\\_iterator\\_tag](#))

- void **\_M\_insert\_aux** (iterator \_\_position, bool \_\_x)
- 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 \_\_x, \_\_true\_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**

### 5.729.1 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 474 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:



- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 5.730 `std::weak_ptr<_Tp>` Class Template Reference

A smart pointer with weak semantics.

Inherits `__weak_ptr<_Tp>`.

### Public Member Functions

- `template<typename _Tp1 > weak_ptr (const shared\_ptr<_Tp1 > &__r)`
- `template<typename _Tp1 > weak_ptr (const weak\_ptr<_Tp1 > &__r)`
- `shared\_ptr<_Tp > lock () const`
- `template<typename _Tp1 > weak\_ptr & operator= (const shared\_ptr<_Tp1 > &__r)`
- `template<typename _Tp1 > weak\_ptr & operator= (const weak\_ptr<_Tp1 > &__r)`

### 5.730.1 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 322 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.731 `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` (const `param_type` &\_\_p)
- `weibull_distribution` (`_RealType` \_\_a=`_RealType`(1), `_RealType` \_\_b=`_RealType`(1))
- `_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` ()

### 5.731.1 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 3796 of file `random.h`.

## 5.731.2 Member Typedef Documentation

### 5.731.2.1 `template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 3803 of file random.h.

## 5.731.3 Member Function Documentation

### 5.731.3.1 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::a () const [inline]`

Return the  $a$  parameter of the distribution.

Definition at line 3850 of file random.h.

### 5.731.3.2 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::b () const [inline]`

Return the  $b$  parameter of the distribution.

Definition at line 3857 of file random.h.

### 5.731.3.3 `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 3886 of file random.h.

### 5.731.3.4 `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 3879 of file random.h.

### 5.731.3.5 `template<typename _RealType = double> void std::weibull_distribution< _RealType >::param (const param_type & __param) [inline]`

Sets the parameter `set` of the distribution.

## 5.731 std::weibull\_distribution<\_RealType> Class Template Reference 3397

### Parameters:

*\_\_param* The new parameter [set](#) of the distribution.

Definition at line 3872 of file random.h.

**5.731.3.6** `template<typename _RealType = double> param_type  
std::weibull_distribution<_RealType>::param() const [inline]`

Returns the parameter [set](#) of the distribution.

Definition at line 3864 of file random.h.

Referenced by `std::operator>>()`.

**5.731.3.7** `template<typename _RealType = double> void  
std::weibull_distribution<_RealType>::reset() [inline]`

Resets the distribution state.

Definition at line 3843 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.732 `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

### 5.732.1 Detailed Description

`template<typename _RealType = double> struct std::weibull_distribution<_RealType>::param_type`

Parameter type.

Definition at line 3805 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

# Chapter 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](#)

#### Defines

- `#define \_GLIBCXX\_PARALLEL\_ALGO\_H`

#### Functions

- `template<typename \_RAIter, typename \_BinaryPredicate >  
\_RAIter std::\_\_parallel::\_\_adjacent\_find\_switch (\_RAIter __begin, \_RAIter  
__end, \_BinaryPredicate __pred, random\_access\_iterator\_tag)`

- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::parallel::adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _FIterator, typename _IteratorTag >`  
`_FIterator std::parallel::adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _RAIter >`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `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 _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 _IteratorTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter __begin, _Iter __end, const _Tp &__value, _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 _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 _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 _IteratorTag1, typename _IteratorTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `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 _Predicate, typename _IteratorTag >`  
`_Iter std::parallel::find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _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 _Tp, typename _IteratorTag >`  
`_Iter std::parallel::find_switch (_Iter __begin, _Iter __end, const _Tp`  
`&__val, _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 _Iter, typename _Function, typename _IteratorTag >`  
`_Function std::parallel::for_each_switch (_Iter __begin, _Iter __end, _`  
`Function __f, _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 _OutputIterator, typename _Size, typename _Generator, typename _`  
`IteratorTag >`  
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __`  
`begin, _Size __n, _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 _Generator, typename _IteratorTag >`  
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end,`  
`_Generator __gen, _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 _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _`  
`FIterator __end, _Compare __comp, _IteratorTag)`
- `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 _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 _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 _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _`  
`FIterator __end, _Compare __comp, _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 _IteratorTag >`  
`_FIterator std::parallel::partition_switch (_FIterator __begin, _FIterator`  
`__end, _Predicate __pred, _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 _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 _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 _Tp, typename _IteratorTag >`  
`void std::parallel::replace_switch (_FIterator __begin, _FIterator __end,`  
`const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `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 _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 _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 _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `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 _RAIter1, typename _RAIter2, typename _OutputRAIter, typename _Predicate >`  
`_OutputRAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputRAIter, typename _Predicate >`  
`_OutputRAIter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_intersection_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputRAIter, typename _Predicate >`  
`_OutputRAIter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_symmetric_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`

- `template<typename _RAIter1 , typename _RAIter2 , typename _OutputRAIter , typename _Predicate >`  
`_OutputRAIter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __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 _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 _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 _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 _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 _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 _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 _FIterator , typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`

- `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 __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __-`  
`end, \_\_gnu\_parallel::sequential\_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 _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, \_\_gnu\_parallel::sequential\_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::\_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::sequential\_tag)`
- `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 >`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`  
`__begin2, _FIterator __end2)`
- `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, 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 _FIterator >`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`  
`__begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`  
`Function __f)`
- `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 _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`  
`Generator __gen)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`  
`__n, _Generator __gen)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `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)`

- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_-`  
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `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, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`__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, 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, \_\_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 _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`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 _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, 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 _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_-`  
`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)`
- `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, \_\_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)`
- `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, \_\_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, typename _BinaryPredicate >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_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 _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 >`  
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_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)`
- `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, typename _Predicate > _OutputIterator std::__parallel::set_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_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_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_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 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_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)`
- `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, \_\_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 > _OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`

- `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 _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::balanced\_quicksort\_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, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `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::parallel\_tag __parallelism)`
- `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)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, _Parallelism __parallelism)`
- `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::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, \_\_gnu\_parallel::balanced\_quicksort\_tag __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::multiway\_mergesort\_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::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_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 _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 _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)`

- `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 _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __-`  
`end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu-`  
`parallel::sequential\_tag)`
- `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)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out, \_\_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](#)

### Defines

- `#define GLIBCXX_PARALLEL_ALGOBASE_H`

### Functions

- `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 _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::__parallel::__lexicographical_compare_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `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 _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _IIter1, _IIter2 > std::__parallel::__mismatch_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`bool std::__parallel::__equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2 >`  
`bool std::__parallel::__equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)`

- `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,`  
`__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)`
- `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, __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, 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, __gnu_parallel::sequential_tag)`

### 6.2.1 Detailed Description

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.

Definition in file `algbase.h`.

## 6.3 algorithm File Reference

### Defines

- `#define _GLIBCXX_ALGORITHM`

### 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`

### Defines

- `#define _EXT_ALGORITHM`

### Functions

- `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 _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > __gnu_cxx::__copy_n (_InputIterator`  
`__first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `int __gnu_cxx::__lexicographical_compare_3way (const char *__first1, const`  
`char *__last1, const char *__first2, const char *__last2)`
- `int __gnu_cxx::__lexicographical_compare_3way (const unsigned char *__-`  
`first1, const unsigned char *__last1, const unsigned char *__first2, const un-`  
`signed char *__last2)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __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 _-`  
`RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator __gnu_cxx::__random_sample (_InputIterator`  
`__first, _InputIterator __last, _RandomAccessIterator __out, _-`  
`RandomNumberGenerator &__rand, const _Distance __n)`
- `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 _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, typename _StrictWeakOrdering >`  
`bool \_\_gnu\_cxx::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_gnu\_cxx::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `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

### Defines

- `#define _PARALLEL_ALGORITHM`

### 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](#)

### Defines

- #define `_GLIBCXX_ALGORITHMFWD_H`

### Functions

- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Filter >`  
`_Filter std::adjacent_find (_Filter, _Filter)`
- `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, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::equal (_Iter1, _Iter1, _Iter2)`

- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _-`  
`Compare)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `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, typename _BinaryPredicate >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `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 _Filter, typename _Generator >`  
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`

- `template<typename _RAIter >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Compare >`  
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter1, typename _Filter2 >`  
`void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _RAIter, typename _Compare >`  
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `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 _Filter, typename _Compare >`  
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`

- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `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 _FIter, typename _Compare >`  
`_FIter std::min_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`  
`_FIter std::min_element (_FIter, _FIter)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `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 >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _FIter, typename _Compare >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2, _BinaryPredicate)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2)`
- `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, typename _Compare >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`

- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > std::partition_copy (_Iter, _Iter, _OIter1, _OIter2,`  
`_Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _RAIter, typename _Compare >`  
`void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`  
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::push_heap (_RAIter, _RAIter)`
- `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 _Iter, typename _OIter, typename _Tp >`  
`_OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::remove_copy_if (_Iter, _Iter, _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 _Iter, typename _OIter, typename _Tp >`  
`_OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter std::replace_copy_if (_Iter, _Iter, _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 _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort_heap (_RAIter, _RAIter)`
- `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 _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation >`  
`_OIter std::transform (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::transform (_IIter, _IIter, _OIter, _UnaryOperation)`
- `template<typename _FIter, typename _BinaryPredicate >`  
`_FIter std::unique (_FIter, _FIter, _BinaryPredicate)`
- `template<typename _FIter >`  
`_FIter std::unique (_FIter, _FIter)`
- `template<typename _IIter, typename _OIter, typename _BinaryPredicate >`  
`_OIter std::unique_copy (_IIter, _IIter, _OIter, _BinaryPredicate)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::unique_copy (_IIter, _IIter, _OIter)`
- `template<typename _FIter, typename _Tp, typename _Compare >`  
`_FIter std::upper_bound (_FIter, _FIter, const _Tp &, _Compare)`
- `template<typename _FIter, typename _Tp >`  
`_FIter std::upper_bound (_FIter, _FIter, const _Tp &)`

### 6.6.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file <bits/algorithmfwd.h>.

## 6.7 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Defines

- `#define \_GLIBCXX\_PARALLEL\_ALGORITHMFWD\_H`

### Functions

- `template<typename \_RAIter , typename \_BiPredicate >  
\_RAIter std::\_\_parallel::\_\_adjacent\_find\_switch (\_RAIter, \_RAIter, \_BiPredicate, random\_access\_iterator\_tag)`
- `template<typename \_FIter , typename \_BiPredicate , typename \_IterTag >  
\_FIter std::\_\_parallel::\_\_adjacent\_find\_switch (\_FIter, \_FIter, \_BiPredicate, \_IterTag)`
- `template<typename \_RAIter >  
\_RAIter std::\_\_parallel::\_\_adjacent\_find\_switch (\_RAIter \_\_begin, \_RAIter \_\_end, random\_access\_iterator\_tag)`
- `template<typename \_FIter , typename \_IterTag >  
\_FIter std::\_\_parallel::\_\_adjacent\_find\_switch (\_FIter, \_FIter, \_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 \_IIter , typename \_Predicate , typename \_IterTag >  
iterator\_traits< \_IIter >::difference\_type std::\_\_parallel::\_\_count\_if\_switch (\_IIter, \_IIter, \_Predicate, \_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 \_IIter , typename \_Tp , typename \_IterTag >  
iterator\_traits< \_IIter >::difference\_type std::\_\_parallel::\_\_count\_switch (\_IIter, \_IIter, const \_Tp &, \_IterTag)`
- `template<typename \_IIter , typename \_FIter , typename \_BiPredicate , typename \_IterTag1 , typename \_IterTag2 >  
\_IIter std::\_\_parallel::\_\_find\_first\_of\_switch (\_IIter, \_IIter, \_FIter, \_FIter, \_BiPredicate, \_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 _IIter, typename _FIter, typename _IterTag1, typename _IterTag2 >`  
`_IIter std::parallel::find_first_of_switch (_IIter, _IIter, _FIter, _FIter, _-`  
`IIterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end,`  
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _IIter, typename _Predicate, typename _IterTag >`  
`_IIter std::parallel::find_if_switch (_IIter, _IIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end,`  
`const _Tp &__val, random_access_iterator_tag)`
- `template<typename _IIter, typename _Tp, typename _IterTag >`  
`_IIter std::parallel::find_switch (_IIter, _IIter, const _Tp &, _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 _IIter, typename _Function, typename _IterTag >`  
`_Function std::parallel::for_each_switch (_IIter, _IIter, _Function, _-`  
`IIterTag)`
- `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 _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _-`  
`IIterTag)`
- `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 _FIter, typename _Generator, typename _IterTag >`  
`void std::parallel::generate_switch (_FIter, _FIter, _Generator, _IterTag)`
- `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 _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`bool std::parallel::lexicographical_compare_switch (_IIter1, _IIter1, _-`  
`IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)`

- `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 _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::max_element_switch (_Filter, _Filter, _Compare, _-`  
`IterTag)`
- `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 _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 _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 _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _-`  
`IterTag)`
- `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, type-`  
`name _IterTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1, _Iter1,`  
`_Iter2, _Predicate, _IterTag1, _IterTag2)`
- `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 _IterTag >`  
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _-`  
`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 _Predicate, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, 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 _Filter, typename _Tp, typename _IterTag >`  
`void std::__parallel::__replace_switch (_Filter, _Filter, const _Tp &, const _Tp`  
`&, _IterTag)`
- `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 _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 _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1,`  
`typename _IterTag2 >`  
`_Filter1 std::__parallel::__search_switch (_Filter1, _Filter1, _Filter2, _Filter2,`  
`_BiPredicate, _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 _IterTag1, typename _IterTag2 >`  
`_Filter1 std::__parallel::__search_switch (_Filter1, _Filter1, _Filter2, _Filter2,`  
`_IterTag1, _IterTag2)`
- `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 _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_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_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_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_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_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_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 _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 _RAIter, typename _RAIter, typename _UnaryOperation >`  
`_RAIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `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 _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, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `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 _Iter , typename _OIter , typename _Predicate , typename _IterTag1 , typename _IterTag2 >`  
`_OIter std::__parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
  - `template<typename _FIter , typename _BiPredicate >`  
`_FIter std::__parallel::adjacent_find (_FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential\_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 >`  
`_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, \_\_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, \_\_gnu\_parallel::sequential\_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::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::sequential\_tag)`
  - `template<typename _Iter , typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
  - `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)`
  - `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,`  
`__gnu_parallel::sequential_tag)`
- `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 _FIter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `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 >`  
`_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)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`  
`__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 __begin, _Iter __end, _Function`  
`__f, __gnu_parallel::sequential_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, __gnu_parallel:: -`  
`Parallelism)`
- `template<typename _FIter, typename _Generator >`  
`void std::__parallel::generate (_FIter, _FIter, _Generator, __gnu -`  
`parallel::sequential_tag)`
- `template<typename _FIter, typename _Generator >`  
`void std::__parallel::generate (_FIter, _FIter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, __gnu -`  
`parallel::Parallelism)`
- `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)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2 >`  
`bool std::__parallel::lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2 >`  
`bool std::__parallel::lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter >`  
`_FIter std::__parallel::max_element (_FIter, _FIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`

- `template<typename _Filter, typename _Compare >`  
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::min_element (_Filter, _Filter)`
- `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, 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, \_\_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, 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, \_\_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 _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _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 _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, 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 _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_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 &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::\_Parallelism)`
- `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 _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)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_BiPredicate)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_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 >`  
`_OIter std::parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `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, \_\_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 _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, \_OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, \_OIter, \_Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `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_symmetric_difference (_IIter1, _IIter1, _IIter2, \_IIter2, \_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, 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 >`  
`_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)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `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 >`  
`_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)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `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::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, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, \_\_gnu\_parallel::sequential\_tag)`
- `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::Parallelism)`

- `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::\_Parallelism)`
- `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)`
- `template<typename _Iter, typename _OIter, typename _Predicate > _OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `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, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter > _OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_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.*

### Namespaces

- namespace `std`

### Defines

- `#define _ALLOCATOR_H`

### Functions

- `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> &)`
- `template<typename _T1, typename _T2>`  
`bool std::operator==(const allocator<_T1> &, const allocator<_T2> &)`

#### 6.8.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [allocator.h](#).



## 6.9 array File Reference

### Defines

- `#define _GLIBCXX_ARRAY`

### 6.9.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

## 6.10 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>`  
`const _Tp & std::get (const array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`_Tp & std::get (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)`

#### 6.10.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/array](#).

## 6.11 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](#)

### Defines

- `#define \_ARRAY\_ALLOCATOR\_H`

### 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 > &)`

#### 6.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array\\_allocator.h](#).

## 6.12 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**

### 6.12.1 Detailed Description

Contains associative containers.

Definition in file [assoc\\_container.hpp](#).

## 6.13 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< void * >`  
*Explicit specialization for void\*.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_ATOMIC`

## Functions

- memory\_order `std::__calculate_memory_order` (memory\_order \_\_m)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool `std::atomic_compare_exchange_strong` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- bool `std::atomic_compare_exchange_strong` (atomic\_address \*\_\_a, void \*\* \_\_v1, void \*\_\_v2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, 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` (atomic\_address \*\_\_a, void \*\* \_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)



- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (__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)`
- `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 (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)`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (__atomic_base< _ITp > *__a, _ITp __i)`
- `bool std::atomic_exchange (atomic_bool *__a, bool __i)`
- `void * std::atomic_exchange (atomic_address *__a, void *__v)`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `bool std::atomic_exchange_explicit (atomic_bool *__a, bool __i, memory_-`  
`order __m)`
- `void * std::atomic_exchange_explicit (atomic_address *__a, void *__v,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`  
`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_explicit (__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_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`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_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m)`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a,`  
`memory_order __m)`
- `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 >`  
`void std::atomic_store (__atomic_base< _ITp > *__a, _ITp __i)`
- `void std::atomic_store (atomic_bool *__a, bool __i)`
- `void std::atomic_store (atomic_address *__a, void *__v)`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void std::atomic_store_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `void std::atomic_store_explicit (atomic_address *__a, void *__v, memory_order __m)`
- `template<typename _Tp >`  
`_Tp std::kill_dependency (_Tp __y)`

### 6.13.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

## 6.14 atomic\_0.h File Reference

### Classes

- struct [\\_\\_atomic0::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic0::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic0::atomic\\_flag](#)  
*atomic\_flag*

### Defines

- #define [\\_ATOMIC\\_CMPEXCHNG](#)(\_\_a, \_\_e, \_\_m, \_\_x)
- #define [\\_ATOMIC\\_LOAD](#)(\_\_a, \_\_x)
- #define [\\_ATOMIC\\_MODIFY](#)(\_\_a, \_\_o, \_\_m, \_\_x)
- #define [\\_ATOMIC\\_STORE](#)(\_\_a, \_\_m, \_\_x)
- #define [\\_GLIBCXX\\_ATOMIC\\_0\\_H](#)

### 6.14.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_0.h](#).

## 6.15 atomic\_2.h File Reference

### Classes

- struct [\\_\\_atomic2::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic2::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic2::atomic\\_flag](#)  
*atomic\_flag*

### Defines

- #define [\\_GLIBCXX\\_ATOMIC\\_2\\_H](#)

#### 6.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_2.h](#).

## 6.16 atomic\_base.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_ATOMIC_CMPEXCHNG_(__a, __e, __m, __x)`
- #define `_ATOMIC_LOAD_(__a, __x)`
- #define `_ATOMIC_MODIFY_(__a, __o, __m, __x)`
- #define `_ATOMIC_STORE_(__a, __m, __x)`
- #define `_GLIBCXX_ATOMIC_BASE_H`
- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

### Typedefs

- typedef struct std::\_\_atomic\_flag\_base **std::\_\_atomic\_flag\_base**
- typedef [atomic\\_short](#) **std::atomic\_int\_fast16\_t**
- typedef [atomic\\_int](#) **std::atomic\_int\_fast32\_t**
- typedef [atomic\\_llong](#) **std::atomic\_int\_fast64\_t**
- typedef [atomic\\_schar](#) **std::atomic\_int\_fast8\_t**
- typedef [atomic\\_short](#) **std::atomic\_int\_least16\_t**
- typedef [atomic\\_int](#) **std::atomic\_int\_least32\_t**
- typedef [atomic\\_llong](#) **std::atomic\_int\_least64\_t**
- typedef [atomic\\_schar](#) **std::atomic\_int\_least8\_t**
- typedef [atomic\\_llong](#) **std::atomic\_intmax\_t**
- typedef [atomic\\_long](#) **std::atomic\_intptr\_t**
- typedef [atomic\\_long](#) **std::atomic\_ptrdiff\_t**
- typedef [atomic\\_ulong](#) **std::atomic\_size\_t**
- typedef [atomic\\_long](#) **std::atomic\_ssize\_t**
- typedef [atomic\\_ushort](#) **std::atomic\_uint\_fast16\_t**
- typedef [atomic\\_uint](#) **std::atomic\_uint\_fast32\_t**
- typedef [atomic\\_ullong](#) **std::atomic\_uint\_fast64\_t**
- typedef [atomic\\_uchar](#) **std::atomic\_uint\_fast8\_t**
- typedef [atomic\\_ushort](#) **std::atomic\_uint\_least16\_t**
- typedef [atomic\\_uint](#) **std::atomic\_uint\_least32\_t**
- typedef [atomic\\_ullong](#) **std::atomic\_uint\_least64\_t**

- typedef [atomic\\_uchar](#) `std::atomic_uint_least8_t`
- typedef [atomic\\_ullong](#) `std::atomic_uintmax_t`
- typedef [atomic\\_ulong](#) `std::atomic_uintptr_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

- void `std::__atomic_flag_wait_explicit` (`__atomic_flag_base *`, `memory_order`)  
    throw ()
- `std::__attribute__` (`((__const__)) __atomic_flag_base *` `__atomic_flag_for_-  
address`(`const void *__z`) throw ()
- void `std::atomic_flag_clear` (`__atomic_flag_base *__a`)
- void `std::atomic_flag_clear_explicit` (`__atomic_flag_base *`, `memory_order`)  
    throw ()
- bool `std::atomic_flag_test_and_set` (`__atomic_flag_base *__a`)
- bool `std::atomic_flag_test_and_set_explicit` (`__atomic_flag_base *`,  
    `memory_order`) throw ()

### 6.16.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic\\_base.h](#).

## 6.17 `atomic_word.h` File Reference

### Defines

- `#define _GLIBCXX_ATOMIC_WORD_H`

### Typedefs

- `typedef int _Atomic_word`

#### 6.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic\\_word.h](#).



## 6.18 atomicfwd\_c.h File Reference

### Defines

- #define **\_ATOMIC\_MEMBER\_**
- #define **atomic\_compare\_exchange**(\_\_a, \_\_e, \_\_m)
- #define **atomic\_compare\_exchange\_explicit**(\_\_a, \_\_e, \_\_m, \_\_x, \_\_y)
- #define **atomic\_exchange**(\_\_a, \_\_m)
- #define **atomic\_exchange\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_add**(\_\_a, \_\_m)
- #define **atomic\_fetch\_add\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_and**(\_\_a, \_\_m)
- #define **atomic\_fetch\_and\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_or**(\_\_a, \_\_m)
- #define **atomic\_fetch\_or\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_sub**(\_\_a, \_\_m)
- #define **atomic\_fetch\_sub\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_xor**(\_\_a, \_\_m)
- #define **atomic\_fetch\_xor\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_is\_lock\_free**(\_\_a)
- #define **atomic\_load**(\_\_a)
- #define **atomic\_load\_explicit**(\_\_a, \_\_x)
- #define **atomic\_store**(\_\_a, \_\_m)
- #define **atomic\_store\_explicit**(\_\_a, \_\_m, \_\_x)

### Typedefs

- typedef struct \_\_atomic\_address\_base **atomic\_address**
- typedef struct \_\_atomic\_bool\_base **atomic\_bool**
- typedef struct \_\_atomic\_char\_base **atomic\_char**
- typedef struct \_\_atomic\_short\_base **atomic\_char16\_t**
- typedef struct \_\_atomic\_int\_base **atomic\_char32\_t**
- typedef struct \_\_atomic\_flag\_base **atomic\_flag**
- typedef struct \_\_atomic\_int\_base **atomic\_int**
- typedef struct \_\_atomic\_llong\_base **atomic\_llong**
- typedef struct \_\_atomic\_long\_base **atomic\_long**
- typedef struct \_\_atomic\_schar\_base **atomic\_schar**
- typedef struct \_\_atomic\_short\_base **atomic\_short**
- typedef struct \_\_atomic\_uchar\_base **atomic\_uchar**
- typedef struct \_\_atomic\_uint\_base **atomic\_uint**
- typedef struct \_\_atomic\_ullong\_base **atomic\_ullong**
- typedef struct \_\_atomic\_ulong\_base **atomic\_ulong**
- typedef struct \_\_atomic\_ushort\_base **atomic\_ushort**
- typedef struct \_\_atomic\_wchar\_t\_base **atomic\_wchar\_t**

### 6.18.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_c.h](#).

## 6.19 atomicfwd\_cxx.h File Reference

### Defines

- #define `_ATOMIC_MEMBER_`

### Typedefs

- 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< long long >` [atomic\\_llong](#)
- typedef `__atomic_base< long >` [atomic\\_long](#)
- typedef `__atomic_base< signed char >` [atomic\\_schar](#)
- typedef `__atomic_base< short >` [atomic\\_short](#)
- typedef `__atomic_base< unsigned char >` [atomic\\_uchar](#)
- typedef `__atomic_base< unsigned int >` [atomic\\_uint](#)
- 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](#)

#### 6.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_cxx.h](#).

## 6.20 atomicity.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define **\_GLIBCXX\_ATOMICITY\_H**
- #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**

#### 6.20.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicity.h](#).

## 6.21 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](#)

### Defines

- `#define \_STL\_AUTO\_PTR\_H`

#### 6.21.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [auto\\_ptr.h](#).

## 6.22 `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`

### Defines

- `#define _GLIBCXX_PARALLEL_BALANCED_QUICKSORT_H`

### 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)`

#### 6.22.1 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](#).

## 6.23 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 >`  
*Similar to `std::multiplies`, but allows two different types.*
- struct `__gnu_parallel::_Plus< _Tp1, _Tp2 >`  
*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`. It features the usual random-access iterator functionality.*
- class `__gnu_parallel::_VoidFunctor< _ValueTp >`  
*Functor that does nothing.*



## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)
- namespace [\\_\\_gnu\\_sequential](#)
- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Defines

- `#define \_GLIBCXX\_PARALLEL\_ASSERT(_Condition)`
- `#define \_GLIBCXX\_PARALLEL\_BASE\_H`

## 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 >  
\_\_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)`

### 6.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

## 6.24 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](#)

### Defines

- `#define _GLIBCXX_PROFILE_BASE_H`

#### 6.24.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

## 6.25 `basic_file.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_BASIC_FILE_STDIO_H`

#### 6.25.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_file.h](#).

## 6.26 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](#)

### Defines

- `#define _BASIC_IOS_H`

### Functions

- `template<typename _Facet >`  
`const _Facet & std::__check_facet (const _Facet *__f)`

#### 6.26.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_ios.h](#).

## 6.27 `basic_ios.tcc` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _BASIC_IOS_TCC`

#### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_ios.tcc](#).

## 6.28 `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.

### Defines

- `#define _GLIBCXX_PARALLEL_BASIC_ITERATOR_H`

### 6.28.1 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](#).

## 6.29 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](#)

### Defines

- #define [\\_BASIC\\_STRING\\_H](#)

### Functions

- [template<>](#)  
[basic\\_istream<wchar\\_t>](#) & [std::getline](#) ([basic\\_istream<wchar\\_t>](#) &\_\_in, [basic\\_string<wchar\\_t>](#) &\_\_str, [wchar\\_t](#) \_\_delim)
- [template<>](#)  
[basic\\_istream<char>](#) & [std::getline](#) ([basic\\_istream<char>](#) &\_\_in, [basic\\_string<char>](#) &\_\_str, [char](#) \_\_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<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 >`  
`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 >`  
`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+ (const 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+ (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 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 >`  
`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 >`  
`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 _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, 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 >`  
`__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 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 >`  
`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 >`  
`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<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is,`  
`basic_string< char > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__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 double \_\_val)
- string **std::to\_string** (double \_\_val)
- string **std::to\_string** (float \_\_val)
- string **std::to\_string** (unsigned long long \_\_val)
- string **std::to\_string** (long long \_\_val)
- string **std::to\_string** (unsigned long \_\_val)
- string **std::to\_string** (long \_\_val)
- string **std::to\_string** (unsigned \_\_val)
- string **std::to\_string** (int \_\_val)
- wstring **std::to\_wstring** (long double \_\_val)
- wstring **std::to\_wstring** (double \_\_val)
- wstring **std::to\_wstring** (float \_\_val)
- wstring **std::to\_wstring** (unsigned long long \_\_val)
- wstring **std::to\_wstring** (long long \_\_val)
- wstring **std::to\_wstring** (unsigned long \_\_val)
- wstring **std::to\_wstring** (long \_\_val)
- wstring **std::to\_wstring** (unsigned \_\_val)
- wstring **std::to\_wstring** (int \_\_val)

### 6.29.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.h](#).

## 6.30 basic\_string.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_BASIC_STRING_TCC`

### Functions

- template `basic_istream< wchar_t > & std::getline` (`basic_istream< wchar_t > &`, `wstring &`)
- template `basic_istream< wchar_t > & std::getline` (`basic_istream< wchar_t > &`, `wstring &`, `wchar_t`)
- template `basic_istream< char > & std::getline` (`basic_istream< char > &`, `string &`)
- template `basic_istream< char > & std::getline` (`basic_istream< char > &`, `string &`, `char`)
- 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 `basic_ostream< wchar_t > & std::operator<<` (`basic_ostream< wchar_t > &`, `const wstring &`)
- template `basic_ostream< char > & std::operator<<` (`basic_ostream< char > &`, `const string &`)
- template `basic_istream< wchar_t > & std::operator>>` (`basic_istream< wchar_t > &`, `wstring &`)
- template `basic_istream< char > & std::operator>>` (`basic_istream< char > &`, `string &`)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc` > `basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _CharT, _Traits > &__is`, `basic_string< _CharT, _Traits, _Alloc > &__str`)

### 6.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.tcc](#).

## 6.31 `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`

#### 6.31.1 Detailed Description

Contains basic types used by containers.

Definition in file `basic_types.hpp`.

## 6.32 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](#)

### Defines

- `#define \_GLIBCXX\_BINDERS\_H`

### 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)`

#### 6.32.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [binders.h](#).

## 6.33 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](#)
- #define [\\_BITMAP\\_ALLOCATOR\\_H](#)

### Enumerations

- enum { [bits\\_per\\_byte](#), [bits\\_per\\_block](#) }

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (size\_t \* \_\_pbitmap, size\_t \_\_pos) throw ()

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (size\_t \*\_\_pbmap, 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 ()

### 6.33.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [bitmap\\_allocator.h](#).

### 6.33.2 Define Documentation

#### 6.33.2.1 #define \_BALLOC\_ALIGN\_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 45 of file [bitmap\\_allocator.h](#).



## 6.34 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`
- `#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<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream<_CharT, _Traits > & std::operator<< (std::basic_ostream<_CharT, _Traits > &__os, const bitset<_Nb > &__x)`
- `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<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)`

### 6.34.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

## 6.35 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)`

#### 6.35.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

## 6.36 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)`

#### 6.36.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

## 6.37 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define **\_BOOST\_CONCEPT\_CHECK\_H**
- #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\_OPERATOR\_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** ()

#### 6.37.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_concept\\_check.h](#).

## 6.38 boost\_sp\_counted\_base.h File Reference

### Classes

- class [std::bad\\_weak\\_ptr](#)  
*Exception possibly thrown by [shared\\_ptr](#).*

### Namespaces

- namespace [std](#)

### Functions

- void [std::\\_\\_throw\\_bad\\_weak\\_ptr\(\)](#)

#### 6.38.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_sp\\_counted\\_base.h](#).

## 6.39 `++0x_warning.h` File Reference

### Defines

- `#define _CXX0X_WARNING_H`

### 6.39.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [++0x\\_warning.h](#).

## 6.40 `cplusplus/allocator.h` File Reference

### Defines

- `#define __glibcxx_base_allocator`
- `#define _GLIBCXX_CXX_ALLOCATOR_H`

### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cplusplus/allocator.h](#).



## 6.41 c++config.h File Reference

### 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\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_BEGIN\_NAMESPACE(X)**
- #define **\_GLIBCXX\_BEGIN\_NESTED\_NAMESPACE(X, Y)**
- #define **\_GLIBCXX\_CONST**
- #define **\_GLIBCXX\_CXX\_CONFIG\_H**
- #define **\_GLIBCXX\_DEPRECATED\_ATTR**
- #define **\_GLIBCXX\_END\_EXTERN\_C**
- #define **\_GLIBCXX\_END\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_END\_NAMESPACE**
- #define **\_GLIBCXX\_END\_NESTED\_NAMESPACE**
- #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_ISINFF`

- #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\_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\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_NAMESPACE\_ASSOCIATION\_VERSION**
- #define **\_GLIBCXX\_NORETURN**
- #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\_PURE**
- #define **\_GLIBCXX\_RES\_LIMITS**

- #define **\_GLIBCXX\_STD**
- #define **\_GLIBCXX\_STD\_D**
- #define **\_GLIBCXX\_STD\_P**
- #define **\_GLIBCXX\_STD\_PR**
- #define **\_GLIBCXX\_STDIO\_MACROS**
- #define **\_GLIBCXX\_SYMVER**
- #define **\_GLIBCXX\_SYMVER\_GNU**
- #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\_ATTR(V)**
- #define **LT\_OBJDIR**
- #define **STDC\_HEADERS**

### 6.41.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++config.h](#).

## 6.42 c++io.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_CXX_IO_H`

### Typedefs

- typedef FILE `std::__c_file`
- typedef `__gthread_mutex_t` `std::__c_lock`

### 6.42.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++io.h](#).

## 6.43 c++locale.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_C_LOCALE_GNU`
- #define `_GLIBCXX_CXX_LOCALE_H`
- #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`,...)

#### 6.43.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale.h](#).

## 6.44 `c++locale_internal.h` File Reference

### 6.44.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale\\_internal.h](#).



## 6.45 `cassert` File Reference

### 6.45.1 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`](#).

## 6.46 ccomplex File Reference

### Defines

- #define `_GLIBCXX_CCOMPLEX`

### 6.46.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

## 6.47 ccomplex File Reference

### Defines

- #define `_GLIBCXX_TR1_CCOMPLEX`

### 6.47.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

## 6.48 cctype File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CCTYPE`

#### 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 `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](#).

## 6.49 ctype File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CCTYPE`

### 6.49.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctype](#).

## 6.50 cctype File Reference

### 6.50.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cctype](#).

## 6.51 `cerrno` File Reference

### Defines

- `#define _GLIBCXX_CERRNO`
- `#define errno`

### 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 `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`](#).

## 6.52 cfenv File Reference

### Defines

- `#define _GLIBCXX_CFENV`

### 6.52.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfenv](#).



## 6.53 cfenv File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CFENV`

### 6.53.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfenv](#).

## 6.54 cfenv File Reference

### 6.54.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cfenv](#).

## 6.55 cfloat File Reference

### Defines

- `#define _GLIBCXX_CFLOAT`
- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.55.1 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](#).

## 6.56 cfloat File Reference

### Defines

- `#define _GLIBCXX_TR1_CFLOAT`

### 6.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

## 6.57 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](#)

### Defines

- [#define \\_CHAR\\_TRAITS\\_EOF](#)
- [#define \\_CHAR\\_TRAITS\\_H](#)

#### 6.57.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [char\\_traits.h](#).

## 6.58 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](#)

### Defines

- #define `_GLIBCXX_PARALLEL_CHECKERS_H`

### Functions

- `template<typename _Iter, typename _Compare >`  
`bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __-`  
`comp)`

#### 6.58.1 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](#).

## 6.59 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, \\_Duration >](#)  
*time\_point*
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)  
*treat\_as\_floating\_point*

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Defines

- `#define \_GLIBCXX\_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 _ToDuration , typename _Rep , typename _Period >`  
`enable_if< __is_duration< _ToDuration >::value, _ToDuration >::type`  
`std::chrono::duration_cast (const duration< _Rep, _Period > &_d)`
- `template<typename _Clock , typename _Duration1 , typename _Duration2 >`  
`bool std::chrono::operator!= (const time_point< _Clock, _Duration1 > &_lhs,`  
`const time_point< _Clock, _Duration2 > &_rhs)`
- `template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >`  
`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< _Rep2, _Rep1 >::type, _Period >`  
`std::chrono::operator* (const _Rep1 &_s, const duration< _Rep2, _Period >`  
`&_d)`
- `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 _Period1 , typename _Clock , typename _Duration2 >`  
`time_point< _Clock, typename common_type< duration< _Rep1, _Period1`  
`>, _Duration2 >::type > std::chrono::operator+ (const duration< _Rep1, _`  
`Period1 > &_lhs, const time_point< _Clock, _Duration2 > &_rhs)`
- `template<typename _Clock , typename _Duration1 , typename _Rep2 , typename _Period2 >`  
`time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`  
`_Period2 > >::type > std::chrono::operator+ (const time_point< _Clock, _`  
`Duration1 > &_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 _Clock , typename _Duration1 , typename _Duration2 >`  
`common_type< _Duration1, _Duration2 >::type std::chrono::operator-`  
`(const time_point< _Clock, _Duration1 > &_lhs, const time_point< _Clock, _`  
`Duration2 > &_rhs)`



- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Duration1 > &__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 _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 _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 _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator< (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator<= (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator== (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator> (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator>= (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`

- `template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >`  
`bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _ToDuration , typename _Clock , typename _Duration >`  
`enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _-`  
`ToDuration > >::type std::chrono::time_point_cast (const time_point< _Clock,`  
`_Duration > &__t)`

### 6.59.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

## 6.60 cinttypes File Reference

### Defines

- #define `_GLIBCXX_CINTTYPES`

### 6.60.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

## 6.61 cinttypes File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CINTTYPES`

### 6.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

## 6.62 cinttypes File Reference

### 6.62.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cinttypes](#).

## 6.63 `ciso646` File Reference

### 6.63.1 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](#).

## 6.64 climits File Reference

### Defines

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.64.1 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](#).

## 6.65 climits File Reference

### Defines

- `#define _GLIBCXX_TR1_CLIMITS`

### 6.65.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).



## 6.66 `locale` File Reference

### Namespaces

- namespace `std`

### Defines

- `#define _GLIBCXX_CLOCALE`

#### 6.66.1 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 `locale`.

## 6.67 cmath File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_CMATH`

### Functions

- `template<typename _Tp > _Tp std::__cmath_power (_Tp, unsigned int)`
- `template<typename _Tp > _Tp std::__pow_helper (_Tp __x, int __n)`
- `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::abs** (float \_\_x)
- double **std::abs** (double \_\_x)
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acos (_Tp __x)`
- long double **std::acos** (long double \_\_x)
- float **std::acos** (float \_\_x)
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asin (_Tp __x)`
- long double **std::asin** (long double \_\_x)
- float **std::asin** (float \_\_x)
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atan (_Tp __x)`
- long double **std::atan** (long double \_\_x)
- float **std::atan** (float \_\_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)`
- long double **std::atan2** (long double \_\_y, long double \_\_x)
- float **std::atan2** (float \_\_y, float \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::ceil (_Tp __x)`
- long double **std::ceil** (long double \_\_x)
- float **std::ceil** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cos (_Tp __x)`
- long double **std::cos** (long double \_\_x)
- float **std::cos** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cosh (_Tp __x)`
- long double **std::cosh** (long double \_\_x)
- float **std::cosh** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::exp (_Tp __x)`
- long double **std::exp** (long double \_\_x)
- float **std::exp** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::fabs (_Tp __x)`
- long double **std::fabs** (long double \_\_x)
- float **std::fabs** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::floor (_Tp __x)`
- long double **std::floor** (long double \_\_x)
- float **std::floor** (float \_\_x)
- long double **std::fmod** (long double \_\_x, long double \_\_y)
- float **std::fmod** (float \_\_x, float \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::frexp (_Tp __x, int *__exp)`
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)
- float **std::frexp** (float \_\_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::ldexp** (long double \_\_x, int \_\_exp)
- float **std::ldexp** (float \_\_x, int \_\_exp)

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
**`std::log`** (`_Tp __x`)
- long double **`std::log`** (long double `__x`)
- float **`std::log`** (float `__x`)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
**`std::log10`** (`_Tp __x`)
- long double **`std::log10`** (long double `__x`)
- float **`std::log10`** (float `__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`)
- long double **`std::pow`** (long double `__x`, long double `__y`)
- float **`std::pow`** (float `__x`, float `__y`)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
**`std::sin`** (`_Tp __x`)
- long double **`std::sin`** (long double `__x`)
- float **`std::sin`** (float `__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`)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
**`std::tanh`** (`_Tp __x`)
- long double **`std::tanh`** (long double `__x`)
- float **`std::tanh`** (float `__x`)

### 6.67.1 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](#).

## 6.68 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- #define `_GLIBCXX_BEGIN_NAMESPACE_TR1`
- #define `_GLIBCXX_END_NAMESPACE_TR1`
- #define `_GLIBCXX_INCLUDE_AS_TR1`
- #define `_GLIBCXX_TR1`
- #define `_GLIBCXX_TR1_CMATH`

### 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, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::tr1::pow** (\_Tp \_\_x, \_Up \_\_y)
- long double **std::tr1::pow** (long double \_\_x, long double \_\_y)
- float **std::tr1::pow** (float \_\_x, float \_\_y)
- double **std::tr1::pow** (double \_\_x, 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`)

### 6.68.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

## 6.69 cmath File Reference

### Namespaces

- namespace [std](#)

### 6.69.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cmath](#).

## 6.70 `cmath.tcc` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CMATH_TCC`

### Functions

- `template<typename _Tp >`  
`_Tp std::__cmath_power (_Tp, unsigned int)`

#### 6.70.1 Detailed Description

This is a Standard C++ Library file.

Definition in file [`cmath.tcc`](#).

## 6.71 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`

### Defines

- `#define _CODECVT_H`

### Functions

- template bool `std::has_facet< codecvt< char, char, mbstate_t > >` (const locale &)
- template bool `std::has_facet< codecvt< wchar_t, char, mbstate_t > >` (const locale &)
- template const `codecvt< char, char, mbstate_t > & std::use_facet< codecvt< char, char, mbstate_t > >` (const locale &)
- template const `codecvt< wchar_t, char, mbstate_t > & std::use_facet< codecvt< wchar_t, char, mbstate_t > >` (const locale &)

**6.71.1 Detailed Description**

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [codecvt.h](#).

## 6.72 `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`

### Defines

- `#define _EXT_CODECVT_SPECIALIZATIONS_H`

### 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)`

#### 6.72.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [codecvt\\_specializations.h](#).

## 6.73 compatibility.h File Reference

### 6.73.1 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](#).

## 6.74 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](#)

### Defines

- #define `_GLIBCXX_PARALLEL_COMPATIBILITY_H`

### 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 ()`

#### 6.74.1 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](#).



## 6.75 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`

### 6.75.1 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](#).

### 6.75.2 Define Documentation

#### 6.75.2.1 `#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()`.

#### 6.75.2.2 `#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 `completetime_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()`.

### 6.75.2.3 `#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 `completetime_settings.h`.

### 6.75.2.4 `#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 `completetime_settings.h`.

### 6.75.2.5 `#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 `completetime_settings.h`.

### 6.75.2.6 `#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 `completetime_settings.h`.

## 6.76 complex File Reference

### Classes

- struct `std::complex<_Tp>`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Defines

- `#define _GLIBCXX_COMPLEX`

### Functions

- `template<typename _Tp >`  
`_Tp std::__complex_abs (const complex<_Tp> &_z)`
- `template<typename _Tp >`  
`_Tp std::__complex_arg (const 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 >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `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::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 > &)`
- `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 >`  
`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 >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`

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

### 6.76.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

## 6.77 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- #define [\\_GLIBCXX\\_BEGIN\\_NAMESPACE\\_TR1](#)
- #define [\\_GLIBCXX\\_END\\_NAMESPACE\\_TR1](#)
- #define [\\_GLIBCXX\\_INCLUDE\\_AS\\_TR1](#)
- #define [\\_GLIBCXX\\_TR1](#)
- #define [\\_GLIBCXX\\_TR1\\_COMPLEX](#)

### Functions

- [template<typename \\_Tp >  
std::complex< typename \\_\\_gnu\\_cxx::\\_\\_promote< \\_Tp >::\\_\\_type >  
std::tr1::conj \(\\_Tp \\_\\_x\)](#)
- [template<typename \\_Tp >  
std::complex< \\_Tp > std::tr1::conj \(const std::complex< \\_Tp > &\\_\\_z\)](#)
- [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 >  
std::complex< \\_Tp > std::tr1::pow \(const std::complex< \\_Tp > &\\_\\_x, const  
std::complex< \\_Tp > &\\_\\_y\)](#)
- [template<typename \\_Tp >  
std::complex< \\_Tp > std::tr1::pow \(const \\_Tp &\\_\\_x, const std::complex< \\_Tp  
> &\\_\\_y\)](#)
- [template<typename \\_Tp >  
std::complex< \\_Tp > std::tr1::pow \(const std::complex< \\_Tp > &\\_\\_x, const  
\\_Tp &\\_\\_y\)](#)

#### 6.77.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

## 6.78 complex File Reference

### Namespaces

- namespace [std](#)

### Functions

- `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 >`  
`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 >`  
`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 >`  
`__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 >`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`



- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `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, 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, 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 >`  
`__gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`

### 6.78.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/complex](#).

## 6.79 complex.h File Reference

### Defines

- `#define _GLIBCXX_COMPLEX_H`

### 6.79.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

## 6.80 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(...)`
- #define `_CONCEPT_CHECK_H`

### 6.80.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concept\\_check.h](#).

## 6.81 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define _CONCURRENCE_H`

### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

### Functions

- void `__gnu_cxx::__throw_concurrency_broadcast_error ()`
- void `__gnu_cxx::__throw_concurrency_lock_error ()`
- void `__gnu_cxx::__throw_concurrency_unlock_error ()`
- void `__gnu_cxx::__throw_concurrency_wait_error ()`

### Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

#### 6.81.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concurrency.h](#).

## 6.82 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`

#### 6.82.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond\\_dealtor.hpp](#).

## 6.83 `condition_variable` File Reference

### Classes

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

### Namespaces

- namespace `std`

### Defines

- `#define _GLIBCXX_CONDITION_VARIABLE`

### Enumerations

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

#### 6.83.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `condition_variable`.

## 6.84 constructors\_destructor\_fn\_imps.hpp File Reference

### Functions

- `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)`
- `PB_DS_CLASS_NAME ()`

#### 6.84.1 Detailed Description

Contains `constructors_destructor_fn_imps` applicable to different containers.

Definition in file [constructors\\_destructor\\_fn\\_imps.hpp](#).

## 6.85 `container_base_dispatch.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.85.1 Detailed Description

Contains an associative container dispatching base.

Definition in file [container\\_base\\_dispatch.hpp](#).



## 6.86 `cpp_type_traits.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Defines

- `#define _CPP_TYPE_TRAITS_H`

#### 6.86.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpp\\_type\\_traits.h](#).

## 6.87 `cpu_defines.h` File Reference

### Defines

- `#define _GLIBCXX_CPU_DEFINES`

### 6.87.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpu\\_defines.h](#).

## 6.88 csetjmp File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSETJMP`
- `#define setjmp(env)`

#### 6.88.1 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](#).

## 6.89 csignal File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSIGNAL`

#### 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 `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](#).

## 6.90 cstdarg File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

#### 6.90.1 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](#).

## 6.91 cstdarg File Reference

### Defines

- `#define _GLIBCXX_TR1_CSTDARG`

### 6.91.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

## 6.92 cstdlib File Reference

### Defines

- `#define _GLIBCXX_CSTDBOOL`

### 6.92.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdlib](#).

## 6.93 cstdint File Reference

### Defines

- `#define _GLIBCXX_TR1_CSTDBOOL`

### 6.93.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).



## 6.94 cstdint File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSTDDEF`

#### 6.94.1 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](#).

## 6.95 cstdint File Reference

### Defines

- `#define _GLIBCXX_CSTDINT`

### 6.95.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

## 6.96 cstdint File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CSTDINT`

### 6.96.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 6.97 cstdint File Reference

### Namespaces

- namespace [std](#)

### 6.97.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdint](#).

## 6.98 cstdio File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSTDIO`

#### 6.98.1 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 [csdio](#).

## 6.99 cstdio File Reference

### Defines

- #define `_GLIBCXX_BEGIN_NAMESPACE_TR1`
- #define `_GLIBCXX_END_NAMESPACE_TR1`
- #define `_GLIBCXX_INCLUDE_AS_TR1`
- #define `_GLIBCXX_TR1`
- #define `_GLIBCXX_TR1_CSTDIO`

### 6.99.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

## 6.100 cstdio File Reference

### Namespaces

- namespace [std](#)

### 6.100.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdio](#).

## 6.101 cstdlib File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_CSTDLIB`
- #define `EXIT_FAILURE`
- #define `EXIT_SUCCESS`

### Functions

- void `std::abort` (void) `_GLIBC_NORETURN` throw ()
- int `std::atexit` (void(\*)()) throw ()
- void `std::exit` (int) `_GLIBC_NORETURN` throw ()

#### 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 `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](#).



## 6.102 cstdlib File Reference

### Defines

- #define `_GLIBCXX_BEGIN_NAMESPACE_TR1`
- #define `_GLIBCXX_END_NAMESPACE_TR1`
- #define `_GLIBCXX_INCLUDE_AS_TR1`
- #define `_GLIBCXX_TR1`
- #define `_GLIBCXX_TR1_CSTDLIB`

### 6.102.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

## 6.103 `cstdlib` File Reference

### 6.103.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdlib](#).

## 6.104 cstring File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CSTRING`

### 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)`

#### 6.104.1 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](#).

## 6.105 `ctgmath` File Reference

### Defines

- `#define _GLIBCXX_CTGMATH`

### 6.105.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).

## 6.106 ctgmath File Reference

### Defines

- #define `_GLIBCXX_TR1_CTGMATH`

### 6.106.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

## 6.107 ctime File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CTIME`

#### 6.107.1 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](#).

## 6.108 ctime File Reference

### Defines

- `#define _GLIBCXX_TR1_CTIME`

### 6.108.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

## 6.109 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)  
*Base class for `ctype`.*

### Namespaces

- namespace [std](#)

#### 6.109.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_base.h](#).



## 6.110 ctype\_inline.h File Reference

### Namespaces

- namespace [std](#)

### 6.110.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_inline.h](#).

## 6.111 ctype\_noninline.h File Reference

### 6.111.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_noninline.h](#).

## 6.112 `cwchar` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CWCHAR`

### Functions

- `wchar_t * std::wcschr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wspbrk` (`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`)

#### 6.112.1 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](#).

## 6.113 cwchar File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CWCHAR`

### 6.113.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

## 6.114 `cwchar` File Reference

### Namespaces

- namespace [std](#)

### 6.114.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cwchar](#).

## 6.115 `cwctype` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CWCTYPE`

#### 6.115.1 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](#).

## 6.116 `cwctype` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_INCLUDE_AS_TR1`
- `#define _GLIBCXX_TR1`
- `#define _GLIBCXX_TR1_CWCTYPE`

### 6.116.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

## 6.117 cwctype File Reference

### Namespaces

- namespace [std](#)

### 6.117.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cwctype](#).



## 6.118 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.*

### Defines

- `#define _CXXABI_FORCED_H`

#### 6.118.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi-forced.h](#).

## 6.119 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](#)

### Defines

- `#define CXXABI_H`
- `#define GLIBCXX_NOTHROW`

### Typedefs

- typedef `__cxa_ctor_return_type(* __cxxabiv1::__cxa_ctor_type)(void *)`

### Functions

- `int __cxxabiv1::__cxa_atexit (void(*)(void *), void *, void *) throw ()`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () __attribute__((__pure__)) throw ()`
- `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 *) throw ()`
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) throw ()`
- `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) throw ()
- **\_\_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)

### 6.119.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

## 6.120 cxxabi\_tweaks.h File Reference

### Defines

- #define `_CXXABI_TWEAKS_H`
- #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\_ -  
\_)))

#### 6.120.1 Detailed Description

The header provides an CPU-variable interface to the C++ ABI.

Definition in file [cxxabi\\_tweaks.h](#).

## 6.121 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\\_MACRO\\_SWITCH\\_H](#)
- #define [\\_GLIBCXX\\_DEBUG\\_ONLY](#)(\_Statement)
- #define [\\_GLIBCXX\\_DEBUG\\_PEDASSERT](#)(\_Condition)

### 6.121.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

## 6.122 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`

### Defines

- `#define _DEBUG_ALLOCATOR_H`

#### 6.122.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. You should only include this header if you are using GCC 3 or later.

Definition in file [debug\\_allocator.h](#).

## 6.123 `debug_map_base.hpp` File Reference

### 6.123.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file [debug\\_map\\_base.hpp](#).

## 6.124 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_DECIMAL`
- `#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** (decimal128 \_\_d)
- double **std::decimal::decimal\_to\_double** (decimal64 \_\_d)
- double **std::decimal::decimal\_to\_double** (decimal32 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal128 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal64 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal32 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal128 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal64 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal32 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal128 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal64 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal32 \_\_d)
- static decimal128 **std::decimal::make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal128 **std::decimal::make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static decimal32 **std::decimal::make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal32 **std::decimal::make\_decimal32** (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!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long \_\_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, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal32 \_\_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 long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)

- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)

- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)

- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)

- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_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, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_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 long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (unsigned int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (decimal64 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (unsigned long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (unsigned int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator<` (decimal32 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator==` (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (unsigned long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (unsigned int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator==` (decimal128 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator==` (decimal128 \_\_lhs, unsigned long \_\_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, decimal128 \_\_rhs)
- `bool std::decimal::operator==` (decimal128 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator==` (decimal128 \_\_lhs, decimal32 \_\_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 long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long \_\_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, decimal128 \_\_rhs)

- `bool std::decimal::operator>` (decimal128 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (decimal128 \_\_lhs, decimal32 \_\_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 long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (unsigned int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (unsigned long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (unsigned int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned long \_\_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, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __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 long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`

### 6.124.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

## 6.125 deque File Reference

### Defines

- `#define _GLIBCXX_DEQUE`

### 6.125.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

## 6.126 deque File Reference

### Classes

- class `std::__debug::deque<_Tp, _Allocator >`  
*Class `std::deque` with safety/checking/debug instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Defines

- `#define _GLIBCXX_DEBUG_DEQUE`

### 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)`

### 6.126.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

## 6.127 deque File Reference

### Classes

- class `std::__profile::deque<_Tp, _Allocator >`  
*Class `std::deque` wrapper with performance instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__profile`

### Defines

- `#define _GLIBCXX_PROFILE_DEQUE`

### 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)`

### 6.127.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).



## 6.128 deque.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_DEQUE_TCC`

### 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)`

### 6.128.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [deque.tcc](#).

## 6.129 `enc_filebuf.h` File Reference

### Classes

- class `__gnu_cxx::enc_filebuf<_CharT>`  
*class `enc_filebuf`.*

### Namespaces

- namespace `__gnu_cxx`

### Defines

- `#define _EXT_ENC_FILEBUF_H`

#### 6.129.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc\\_filebuf.h](#).

## 6.130 `equally_split.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- `#define _GLIBCXX_PARALLEL_EQUALLY_SPLIT_H`

### 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)`

#### 6.130.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [equally\\_split.h](#).

## 6.131 error\_constants.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_ERROR_CONSTANTS`

### 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 }
```

### 6.131.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [error\\_constants.h](#).

## 6.132 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\_\_))

#### 6.132.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

## 6.133 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)

#### 6.133.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

## 6.134 `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 ()`
- `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\_\_\(\(\_\_noreturn\_\_\)\)`

#### 6.134.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [exception\\_ptr.h](#).



## 6.135 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`

### Defines

- `#define _EXTPTR_ALLOCATOR_H`

### Functions

- `template<typename _Tp>`  
`void __gnu_cxx::swap (_ExtPtr_allocator<_Tp> &__larg, _ExtPtr_allocator<_Tp> &__rarg)`

#### 6.135.1 Detailed Description

##### 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](#).

## 6.136 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_PARALLEL_FEATURES_H`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 6.136.1 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](#).

### 6.136.2 Define Documentation

#### 6.136.2.1 `#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](#).

#### 6.136.2.2 `#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](#).

**6.136.2.3 #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`.

**6.136.2.4 #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`.

**6.136.2.5 #define \_GLIBCXX\_MERGESORT**

Include parallel multi-way mergesort.

**See also:**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

**6.136.2.6 #define \_GLIBCXX\_QUICKSORT**

Include parallel unbalanced quicksort.

**See also:**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

**6.136.2.7 #define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iiter beg, _Iiter __end)`.

**See also:**

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file features.h.

**6.136.2.8 #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.

**6.136.2.9 #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.

## 6.137 fenv.h File Reference

### Defines

- #define `_GLIBCXX_FENV_H`

### 6.137.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fenv.h](#).

## 6.138 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](#)

### Defines

- `#define _GLIBCXX_PARALLEL_FIND_H`

### 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, 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)`
- `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)`

#### 6.138.1 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](#).

## 6.139 `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`

### Defines

- `#define _GLIBCXX_PARALLEL_FIND_SELECTORS_H`

#### 6.139.1 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](#).

## 6.140 for\_each.h File Reference

Main interface for embarrassingly parallel functions.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- #define `_GLIBCXX_PARALLEL_FOR_EACH_H`

### 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)`

#### 6.140.1 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](#).



## 6.141 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](#)

## Defines

- #define [\\_GLIBCXX\\_PARALLEL\\_FOR\\_EACH\\_SELECTORS\\_H](#)

### 6.141.1 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](#).

## 6.142 formatter.h File Reference

### Classes

- struct [\\_\\_gnu\\_debug::\\_\\_is\\_same<\\_Type1, \\_Type2 >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Defines

- `#define` [\\_GLIBCXX\\_DEBUG\\_FORMATTER\\_H](#)

### 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](#) }

### 6.142.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

## 6.143 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, _Alloc >`  
*A forward\_list::const\_iterator.*
- struct `std::_Fwd_list_iterator<_Tp, _Alloc >`  
*A forward\_list::iterator.*
- struct `std::_Fwd_list_node<_Tp, _Alloc >`  
*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<_Alloc >`  
*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`

### Defines

- `#define FORWARD_LIST_H`

### Functions

- `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 _Fwd_list_iterator<_Tp, _Alloc > &__x, const _Fwd_list_const_iterator<_Tp, _Alloc > &__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 >`  
`bool std::operator== (const Fwd_list_iterator< _Tp, _Alloc > &__x, const Fwd_list_const_iterator< _Tp, _Alloc > &__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 >`  
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`

### 6.143.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.h](#).

## 6.144 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_FORWARD_LIST_TCC`

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &_lx, const`  
`forward_list< _Tp, _Alloc > &_ly)`

#### 6.144.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.tcc](#).

## 6.145 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](#)

### Defines

- `#define \_GLIBCXX\_FSTREAM`

#### 6.145.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fstream](#).



## 6.146 `fstream.tcc` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _FSTREAM_TCC`

#### 6.146.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [fstream.tcc](#).

## 6.147 funtexcept.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _FUNCTEXCEPT_H`

### 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__ -  
))`

### 6.147.1 Detailed Description

This header provides support for `-fno-exceptions`.

Definition in file [funtexcept.h](#).

## 6.148 functional File Reference

### Classes

- struct `std::_is_location_invariant<_Tp>`
- struct `std::_Build_index_tuple<_Num>`  
*Builds an `_Index_tuple`<0, 1, 2, ..., `_Num-1`>.*
- 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.*
- class `std::_Has_result_type_helper<_Tp>`
- struct `std::_Index_tuple<_Indexes>`
- 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`

## Defines

- `#define _GLIBCXX_FUNCTIONAL`

## Enumerations

- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }

## Functions

- `template<typename _Member , typename _Class >`  
`_Mem_fn< _Member _Class::* > std::callable_function (_Member _-`  
`Class::*const &_p)`
- `template<typename _Member , typename _Class >`  
`_Mem_fn< _Member _Class::* > std::callable_function (_Member _-`  
`Class::*&_p)`
- `template<typename _Function >`  
`_Function & std::callable_function (_Function &_f)`
- `template<typename _Function , typename... _Args>`  
`enable_if< (is_pointer< _Function >::value &&is_function< typename`  
`remove_pointer< _Function >::type >::value), typename result_of< _Function(_-`  
`Args...)>::type >::type std::invoke (_Function _f, _Args &&..._args)`
- `template<typename _Function , typename... _Args>`  
`enable_if< (!is_member_pointer< _Function >::value &&!is_function< _-`  
`Function >::value &&!is_function< typename remove_pointer< _Function`  
`>::type >::value), typename result_of< _Function(_Args...)>::type >::type`  
`std::invoke (_Function &_f, _Args &&..._args)`
- `template<typename _Result , typename _Function , typename... _ArgTypes>`  
`_Bind_result< _Result, typename _Maybe_wrap_member_pointer< _Function`  
`>::type(_ArgTypes...)> std::bind (_Function _f, _ArgTypes..._args)`
- `template<typename _Function , typename... _ArgTypes>`  
`_Bind< typename _Maybe_wrap_member_pointer< _Function >::type(_-`  
`ArgTypes...)> std::bind (_Function _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!= (_M_clear_type *, const function< _Res(_Args...)> &_f)`
- `template<typename _Res , typename... _Args>`  
`bool std::operator!= (const function< _Res(_Args...)> &_f, _M_clear_type *)`
- `template<typename _Res , typename... _Args>`  
`bool std::operator== (_M_clear_type *, const function< _Res(_Args...)> &_f)`
- `template<typename _Res , typename... _Args>`  
`bool std::operator== (const function< _Res(_Args...)> &_f, _M_clear_type *)`
- `template<typename _Res , typename... _Args>`  
`void std::swap (function< _Res(_Args...)> &_x, function< _Res(_Args...)>`  
`&_y)`
  
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (reference_wrapper< _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< _Tp > std::ref (_Tp &__t)`

### 6.148.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

## 6.149 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`

### Defines

- `#define _EXT_FUNCTIONAL`



## 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))`

### 6.149.1 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 [ext/functional](#).

## 6.150 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)`
- `#define \_FUNCTIONAL\_HASH\_H`

#### 6.150.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [functional\\_hash.h](#).

## 6.151 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Defines

- #define `_GLIBCXX_DEBUG_FUNCTIONS_H`

### Functions

- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Safe_iterator< _Iterator,`  
`_Sequence > &__x)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (_Iterator &)`
- `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 >`  
`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\_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, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Safe_iterator< _Iterator, _-`  
`Sequence > &__x)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Tp *__ptr)`
- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_singular (_Iterator &__x)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`
- `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 (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, 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 >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _-`  
`InputIterator &, std::input\_iterator\_tag)`
- `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 _InputIterator1, typename _InputIterator2 >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::\_\_false\_type)`
- `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 &, const _-`  
`InputIterator &, std::\_\_false\_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, std::\_\_true\_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)

### 6.151.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).

## 6.152 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_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.*
- 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`

## Defines

- `#define _GLIBCXX_FUTURE`

## Enumerations

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

## Functions

- `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)`

- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::type > std::async (launch __-`  
`policy, _Fn &&__fn, _Args &&...__args)`
- `error_code std::make_error_code (future_errc __errc)`
- `error_condition std::make_error_condition (future_errc __errc)`
- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task<`  
`_Res(_ArgTypes...)> &__y)`
- `template<typename _Res >`  
`void std::swap (promise< _Res > &__x, promise< _Res > &__y)`

## Variables

- `const error_category *const std::future\_category`

### 6.152.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).



## 6.153 `gslice.h` File Reference

### Classes

- class [std::gslice](#)  
*Class defining multi-dimensional subset of an [array](#).*

### Namespaces

- namespace [std](#)

### Defines

- `#define _GSLICE_H`

#### 6.153.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice.h](#).

## 6.154 `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)`
- `#define _GSLICE_ARRAY_H`

#### 6.154.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice\\_array.h](#).

## 6.155 hash\_fun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_HASH_FUN_H`

### Functions

- `size_t __gnu_cxx::__stl_hash_string` (const char \* \_\_s)

#### 6.155.1 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](#).

## 6.156 hash\_map File Reference

### Classes

- class `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc >`
- class `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqKey, _Alloc >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Defines

- `#define _HASH_MAP`

### Functions

- `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 >`  
`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 _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)`
- `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)`

### 6.156.1 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](#).

## 6.157 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**

### 6.157.1 Detailed Description

Contains hash-related policies.

Definition in file [hash\\_policy.hpp](#).

## 6.158 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](#)

### Defines

- `#define \_HASH\_SET`

### Functions

- `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 _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 >`  
`void \_\_gnu\_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, 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)`

### 6.158.1 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](#).



## 6.159 hashtable.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_HASHTABLE_H`

### 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`]

#### 6.159.1 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](#).

## 6.160 `hashtable.h` File Reference

### 6.160.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/hashtable.h](#).

## 6.161 hashtable\_policy.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Defines

- #define [\\_HASHTABLE\\_POLICY\\_H](#)

### Functions

- [template<class \\_Iterator > std::iterator\\_traits< \\_Iterator >::difference\\_type std::\\_\\_detail::\\_\\_distance\\_fw \(\\_Iterator \\_\\_first, \\_Iterator \\_\\_last\)](#)
- [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 \(\\_Iterator \\_\\_first, \\_Iterator \\_\\_last, std::input\\_iterator\\_tag\)](#)
- [template<typename \\_RAIter, typename \\_Tp > \\_RAIter std::\\_\\_detail::\\_\\_lower\\_bound \(\\_RAIter \\_\\_first, \\_RAIter \\_\\_last, const \\_Tp &\\_\\_val\)](#)
- [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!= \(const \\_Node\\_iterator\\_base< \\_Value, \\_\\_cache > &\\_\\_x, const \\_Node\\_iterator\\_base< \\_Value, \\_\\_cache > &\\_\\_y\)](#)
- [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 \[\]](#)

### 6.161.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [hashtable\\_policy.h](#).

## 6.162 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)`
- `#define _INDIRECT_ARRAY_H`

#### 6.162.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [indirect\\_array.h](#).

## 6.163 `initializer_list` File Reference

### Classes

- class `std::initializer_list<_E>`  
*[initializer\\_list](#)*

### Namespaces

- namespace `std`

#### 6.163.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer\\_list](#).

## 6.164 iomanip File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_IOMANIP`

### Functions

- `template<typename _MoneyT >`  
`_Get_money< _MoneyT >` [std::get\\_money](#) (`_MoneyT &__mon`, `bool __intl=false`)
- `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`, `_Setw __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_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`, `_Setbase __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`, `_Resetiosflags __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`, `_Setw __f`)

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setprecision __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, _Setbase __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, _Resetiosflags __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)`

### 6.164.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).



## 6.165 ios File Reference

### Defines

- `#define _GLIBCXX_IOS`

### 6.165.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

## 6.166 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](#)

### Defines

- `#define _IOS_BASE_H`
- `#define _IOS_BASE_SEEK_CUR`
- `#define _IOS_BASE_SEEK_END`

### 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)
- \_Ios\_Iostate **std::operator&** (\_Ios\_Iostate \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode **std::operator&** (\_Ios\_Openmode \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags **std::operator&** (\_Ios\_Fmtflags \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate & **std::operator&=** (\_Ios\_Iostate & \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode & **std::operator&=** (\_Ios\_Openmode & \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags & **std::operator&=** (\_Ios\_Fmtflags & \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate **std::operator^** (\_Ios\_Iostate \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode **std::operator^** (\_Ios\_Openmode \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags **std::operator^** (\_Ios\_Fmtflags \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate & **std::operator^=** (\_Ios\_Iostate & \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode & **std::operator^=** (\_Ios\_Openmode & \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags & **std::operator^=** (\_Ios\_Fmtflags & \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate **std::operator|** (\_Ios\_Iostate \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode **std::operator|** (\_Ios\_Openmode \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags **std::operator|** (\_Ios\_Fmtflags \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate & **std::operator|=** (\_Ios\_Iostate & \_\_a, \_Ios\_Iostate \_\_b)
- \_Ios\_Openmode & **std::operator|=** (\_Ios\_Openmode & \_\_a, \_Ios\_Openmode \_\_b)
- \_Ios\_Fmtflags & **std::operator|=** (\_Ios\_Fmtflags & \_\_a, \_Ios\_Fmtflags \_\_b)
- \_Ios\_Iostate **std::operator~** (\_Ios\_Iostate \_\_a)
- \_Ios\_Openmode **std::operator~** (\_Ios\_Openmode \_\_a)
- \_Ios\_Fmtflags **std::operator~** (\_Ios\_Fmtflags \_\_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`)

### 6.166.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ios\\_base.h](#).

## 6.167 iosfwd File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_IOSFWD`

### 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](#)

### 6.167.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

## 6.168 iostream File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_GLIBCXX_IOSTREAM`

### 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.

- ostream [std::cerr](#)
- istream [std::cin](#)
- ostream [std::clog](#)
- ostream [std::cout](#)
- wostream [std::wcerr](#)
- wistream [std::wcin](#)
- wostream [std::wclog](#)
- wostream [std::wcout](#)

### 6.168.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).

## 6.169 istream File Reference

### Classes

- 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\\_istream< \\_CharT, \\_Traits >::sentry](#)  
*Performs setup work for input streams.*

### Namespaces

- namespace [std](#)

### Defines

- `#define \_GLIBCXX\_ISTREAM`

### 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<class _Traits >  
basic\_istream< char, \_Traits > & std::operator>> (basic\_istream< char, \_Traits > &__in, signed char *__s)`
- `template<class _Traits >  
basic\_istream< char, \_Traits > & std::operator>> (basic\_istream< char, \_Traits > &__in, unsigned char *__s)`

- `template<>`  
`basic_istream< char > & std::operator>>` (`basic_istream< char > &__in,`  
`char *__s`)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _-`  
`CharT, _Traits > &__in, _CharT *__s`)
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>>` (`basic_istream< char, _-`  
`Traits > &__in, signed char &__c`)
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>>` (`basic_istream< char, _-`  
`Traits > &__in, unsigned char &__c`)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _-`  
`CharT, _Traits > &__in, _CharT &__c`)

### 6.169.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).



## 6.170 istream.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _ISTREAM_TCC`

### Functions

- `template wistream & std::operator>> (wistream &, wchar_t *)`
- `template wistream & std::operator>> (wistream &, wchar_t &)`
- `template istream & std::operator>> (istream &, signed char *)`
- `template istream & std::operator>> (istream &, unsigned char *)`
- `template istream & std::operator>> (istream &, signed char &)`
- `template istream & std::operator>> (istream &, unsigned char &)`
- `template istream & std::operator>> (istream &, char *)`
- `template istream & std::operator>> (istream &, char &)`
- `template istream & std::ws (istream &)`
- `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 *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__in, _CharT &__c)`

### 6.170.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [istream.tcc](#).

## 6.171 iterator File Reference

### Defines

- `#define _GLIBCXX_ITERATOR`

### 6.171.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

## 6.172 iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define [\\_EXT\\_ITERATOR](#)

### Functions

- `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, std::input\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _-`  
`Distance &__n)`

#### 6.172.1 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](#).

## 6.173 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](#)

### Defines

- `#define _GLIBCXX_PARALLEL_ITERATOR_H`

#### 6.173.1 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](#).

## 6.174 iterator\_tracker.h File Reference

### Namespaces

- namespace `std`
- namespace `std::__profile`

### Defines

- `#define GLIBCXX_PROFILE_ITERATOR_TRACKER`

### Functions

- `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 >`  
`__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 _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 _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 _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)`

### 6.174.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator\\_tracker.h](#).

## 6.175 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)`
- #define `_GLIBCXX_NUMERIC_LIMITS`

## Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` {  
    `std::round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`,  
    `std::round_toward_infinity`,  
    `std::round_toward_neg_infinity` }

### 6.175.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

## 6.176 list File Reference

### Defines

- `#define _GLIBCXX_LIST`

### 6.176.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).

## 6.177 list File Reference

### Classes

- class `std::__debug::list< _Tp, _Allocator >`  
*Class `std::list` with safety/checking/debug instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Defines

- `#define _GLIBCXX_DEBUG_LIST`

### 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)`

### 6.177.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

## 6.178 list File Reference

### Classes

- class `std::__profile::list<_Tp, _Allocator >`  
*List wrapper with performance instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__profile`

### Defines

- `#define _GLIBCXX_PROFILE_LIST`

### 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)`

### 6.178.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

## 6.179 list.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _LIST_TCC`

### 6.179.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [list.tcc](#).

## 6.180 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](#)

### Defines

- #define `_GLIBCXX_PARALLEL_LIST_PARTITION_H`

### 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 _FuncType >`  
`size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FuncType &__f, int __oversampling=0)`

#### 6.180.1 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](#).



## 6.181 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`

### 6.181.1 Detailed Description

Contains policies for list update containers.

Definition in file [list\\_update\\_policy.hpp](#).

## 6.182 locale File Reference

### Defines

- `#define _GLIBCXX_LOCALE`

### 6.182.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

## 6.183 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`

### Defines

- `#define _LOCALE_CLASSES_H`

### Functions

- `template<typename _Facet >`  
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use_facet (const locale &__loc)`

### 6.183.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_classes.h](#).

## 6.184 locale\_classes.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _LOCALE_CLASSES_TCC`

### Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template bool std::has\_facet< collate< char > > (const locale &)`
- `template bool std::has\_facet< collate< wchar_t > > (const locale &)`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`
- `template const collate< char > & std::use\_facet< collate< char > > (const locale &)`
- `template const collate< wchar_t > & std::use\_facet< collate< wchar_t > > (const locale &)`

#### 6.184.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_classes.tcc](#).

## 6.185 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`
- `#define _LOCALE_FACETS_H`

## 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<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, 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<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter std::__write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT >`  
`ostreambuf_iterator<_CharT> std::__write (ostreambuf_iterator<_CharT> __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::isctrl (_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)`

### 6.185.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.h](#).



## 6.186 locale\_facets.tcc File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `_LOCALE_FACETS_TCC`

### Functions

- `template<typename _CharT > _CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `std::__attribute__ ((__pure__)) bool __verify_grouping(const char * __grouping)`
- `template<typename _CharT, typename _ValueT > int std::__int_to_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios_base::fmtflags __flags, bool __dec)`
- `template bool std::has_facet< ctype< char > > (const locale &)`
- `template bool std::has_facet< ctype< wchar_t > > (const locale &)`
- `template bool std::has_facet< num_get< char > > (const locale &)`
- `template bool std::has_facet< num_get< wchar_t > > (const locale &)`
- `template bool std::has_facet< num_put< char > > (const locale &)`
- `template bool std::has_facet< num_put< wchar_t > > (const locale &)`
- `template bool std::has_facet< numpunct< char > > (const locale &)`
- `template bool std::has_facet< numpunct< wchar_t > > (const locale &)`
- `size_t const string & __grouping_tmp std::throw ()`
- `template const ctype< char > & std::use_facet< ctype< char > > (const locale &)`
- `template const ctype< wchar_t > & std::use_facet< ctype< wchar_t > > (const locale &)`
- `template const num_get< char > & std::use_facet< num_get< char > > (const locale &)`
- `template const num_get< wchar_t > & std::use_facet< num_get< wchar_t > > (const locale &)`
- `template const num_put< char > & std::use_facet< num_put< char > > (const locale &)`
- `template const num_put< wchar_t > & std::use_facet< num_put< wchar_t > > (const locale &)`
- `template const numpunct< char > & std::use_facet< numpunct< char > > (const locale &)`

- `template const numpunct< wchar_t > & std::use_facet< numpunct< wchar_t > > (const locale &)`

## Variables

- `size_t std::__grouping_size`

### 6.186.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.tcc](#).

## 6.187 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](#)

## Defines

- `#define \_LOCALE\_FACETS\_NONIO\_H`

### 6.187.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.h](#).

## 6.188 locale\_facets\_nonio.tcc File Reference

### Namespaces

- namespace `std`

### Defines

- `#define _LOCALE_FACETS_NONIO_TCC`

### Functions

- template bool `std::has_facet< __timepunct< char > >` (const locale &)
- template bool `std::has_facet< __timepunct< wchar_t > >` (const locale &)
- template bool `std::has_facet< messages< char > >` (const locale &)
- template bool `std::has_facet< messages< wchar_t > >` (const locale &)
- template bool `std::has_facet< money_get< char > >` (const locale &)
- template bool `std::has_facet< money_get< wchar_t > >` (const locale &)
- template bool `std::has_facet< money_put< char > >` (const locale &)
- template bool `std::has_facet< money_put< wchar_t > >` (const locale &)
- template bool `std::has_facet< moneypunct< char > >` (const locale &)
- template bool `std::has_facet< moneypunct< wchar_t > >` (const locale &)
- template bool `std::has_facet< time_get< char > >` (const locale &)
- template bool `std::has_facet< time_get< wchar_t > >` (const locale &)
- template bool `std::has_facet< time_put< char > >` (const locale &)
- template bool `std::has_facet< time_put< wchar_t > >` (const locale &)
- template const `__timepunct< char > & std::use_facet< __timepunct< char > >` (const locale &)
- template const `__timepunct< wchar_t > & std::use_facet< __timepunct< wchar_t > >` (const locale &)
- template const `messages< char > & std::use_facet< messages< char > >` (const locale &)
- template const `messages< wchar_t > & std::use_facet< messages< wchar_t > >` (const locale &)
- template const `money_get< char > & std::use_facet< money_get< char > >` (const locale &)
- template const `money_get< wchar_t > & std::use_facet< money_get< wchar_t > >` (const locale &)
- template const `money_put< char > & std::use_facet< money_put< char > >` (const locale &)
- template const `money_put< wchar_t > & std::use_facet< money_put< wchar_t > >` (const locale &)

- `template const moneypunct< char, false > & std::use_facet< moneypunct< char, false > > (const locale &)`
- `template const moneypunct< char, true > & std::use_facet< moneypunct< char, true > > (const locale &)`
- `template const moneypunct< wchar_t, false > & std::use_facet< moneypunct< wchar_t, false > > (const locale &)`
- `template const moneypunct< wchar_t, true > & std::use_facet< moneypunct< wchar_t, true > > (const locale &)`
- `template const time_get< char > & std::use_facet< time_get< char > > (const locale &)`
- `template const time_get< wchar_t > & std::use_facet< time_get< wchar_t > > (const locale &)`
- `template const time_put< char > & std::use_facet< time_put< char > > (const locale &)`
- `template const time_put< wchar_t > & std::use_facet< time_put< wchar_t > > (const locale &)`

### 6.188.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.tcc](#).

## 6.189 localefd.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _LOCALE_FWD_H`

### 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::isctrl (_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)`

### 6.189.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [localefwd.h](#).



## 6.190 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](#)

## Defines

- #define [\\_GLIBCXX\\_PARALLEL\\_LOSERTREE\\_H](#)

### 6.190.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

## 6.191 macros.h File Reference

### Defines

- #define `__glibcxx_check_erase`(\_Position)
- #define `__glibcxx_check_erase_range`(\_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_range`(\_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_MACROS_H`
- #define `_GLIBCXX_DEBUG_VERIFY`(\_Condition, \_ErrorMessage)

### 6.191.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

### 6.191.2 Define Documentation

#### 6.191.2.1 #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 102 of file `macros.h`.

**6.191.2.2 #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 116 of file macros.h.

**6.191.2.3 #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 229 of file macros.h.

**6.191.2.4 #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 (i.e., 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.

**6.191.2.5 #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 87 of file macros.h.

**6.191.2.6 #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 177 of file macros.h.

**6.191.2.7 #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 197 of file macros.h.

**6.191.2.8 #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 209 of file macros.h.

**6.191.2.9 #define \_\_glibcxx\_check\_sorted\_pred(\_First, \_Last, \_Pred)**

Verify that the iterator range [\_First, \_Last) is sorted by the predicate \_Pred.

Definition at line 148 of file macros.h.

**6.191.2.10 #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--()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

## 6.192 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](#)

### Defines

- `#define _MALLOC_ALLOCATOR_H`

### 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 > &)`

#### 6.192.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc\\_allocator.h](#).

## 6.193 map File Reference

### Defines

- `#define _GLIBCXX_MAP`

### 6.193.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

## 6.194 map File Reference

### Defines

- `#define _GLIBCXX_DEBUG_MAP`

### 6.194.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).



## 6.195 map File Reference

### Defines

- `#define _GLIBCXX_PROFILE_MAP`

### 6.195.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

## 6.196 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`

### Defines

- `#define _GLIBCXX_DEBUG_MAP_H`

### 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)`

### **6.196.1 Detailed Description**

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

## 6.197 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`

### Defines

- `#define _GLIBCXX_PROFILE_MAP_H`

### 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)`

### 6.197.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

## 6.198 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)`
- `#define _MASK_ARRAY_H`

#### 6.198.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [mask\\_array.h](#).

## 6.199 memory File Reference

### Defines

- `#define _GLIBCXX_MEMORY`

### 6.199.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [memory](#).

## 6.200 memory File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_EXT_MEMORY`

### Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::\_\_uninitialized\_copy\_n (_-`  
`InputIter __first, _Size __count, _ForwardIter __result)`
- `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, std::input\_iterator\_tag)`
- `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)`

#### 6.200.1 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](#).

## 6.201 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](#)

### Defines

- `#define` `_GLIBCXX_PARALLEL_MERGE_H`

### 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`)

### 6.201.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

## 6.202 messages\_members.h File Reference

### Namespaces

- namespace [std](#)

### 6.202.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [messages\\_members.h](#).

## 6.203 move.h File Reference

### Classes

- struct [std::identity](#)< \_Tp >  
*identity*

### Namespaces

- namespace [std](#)

### Defines

- #define [\\_GLIBCXX\\_FORWARD](#)(\_Tp, \_\_val)
- #define [\\_GLIBCXX\\_MOVE](#)(\_Tp)
- #define [\\_MOVE\\_H](#)

### Functions

- [template](#)<typename \_Tp >  
[enable\\_if](#)< [is\\_lvalue\\_reference](#)< \_Tp >::value, \_Tp >::type [std::forward](#)  
(typename [std::remove\\_reference](#)< \_Tp >::type &&\_\_t)
- [template](#)<typename \_Tp >  
[enable\\_if](#)< [is\\_lvalue\\_reference](#)< \_Tp >::value, \_Tp >::type [std::forward](#)  
(typename [std::identity](#)< \_Tp >::type \_\_t)
- [template](#)<typename \_Tp >  
[enable\\_if](#)<![is\\_lvalue\\_reference](#)< \_Tp >::value, \_Tp && >::type [std::forward](#)  
(typename [std::identity](#)< \_Tp >::type &&\_\_t)
- [template](#)<typename \_Tp >  
[enable\\_if](#)<![is\\_lvalue\\_reference](#)< \_Tp >::value, \_Tp && >::type [std::forward](#)  
(typename [std::identity](#)< \_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)

### 6.203.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [move.h](#).

## 6.204 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`
- `#define _MT_ALLOCATOR_H`

### 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 > &)`

### 6.204.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt\\_allocator.h](#).



## 6.205 `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`

### Defines

- `#define _GLIBCXX_DEBUG_MULTIMAP_H`

### 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)`

### 6.205.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

## 6.206 `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`

### Defines

- `#define _GLIBCXX_PROFILE_MULTIMAP_H`

### 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, _-`  
`_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, _-`  
`_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, _-`  
`_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, _-`  
`_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, _-`  
`_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, _-`  
`_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)`

### 6.206.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

## 6.207 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)`
- `#define _GLIBCXX_PARALLEL_MULTISEQ_SELECTION_H`

### 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 >())`

#### 6.207.1 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](#).

## 6.208 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](#)

### Defines

- `#define \_GLIBCXX\_DEBUG\_MULTISSET\_H`

### 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)`

### 6.208.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).



## 6.209 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](#)

### Defines

- `#define \_GLIBCXX\_PROFILE\_MULTISSET\_H`

### 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)`

### 6.209.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

## 6.210 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, `default_parallel_tag` \_\_tag)
- 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, `parallel_tag` \_\_tag=`parallel_tag(0)`)
- 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::sampling_tag` \_\_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, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_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< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `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)`

- `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 >`  
`_RAIterOut \_\_gnu\_parallel::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 , type-`  
`name _Compare >`  
`_RAIterOut \_\_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 >`  
`_RAIterOut \_\_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 >`  
`_RAIterOut \_\_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 >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator _-`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , type-`  
`name _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)`
- `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, parallel_tag __tag=parallel_tag(0))

```

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`__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, typename _Compare >`  
`__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, typename _Compare >`  
`__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 >`  
`__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 >`  
`__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 >`  
`__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 >`  
`__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)

```

### 6.210.1 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](#).

### 6.210.2 Define Documentation

#### 6.210.2.1 #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\(\)](#).



## 6.211 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::\\_PMWMSortingData<\\_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](#)

### Defines

- `#define \_GLIBCXX\_PARALLEL\_MULTIWAY\_MERGESORT\_H`

### Functions

- `template<typename _RAIter, typename _DifferenceTp >  
void \_\_gnu\_parallel::\_\_determine\_samples (_PMWMSortingData<_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 (_PMWMSortingData< _-`  
`RAIter > *__sd, _Compare &__comp)`

### 6.211.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_mergesort.h](#).

## 6.212 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`

### Defines

- `#define _GLIBCXX_MUTEX`

## Functions

- mutex & **std::\_\_get\_once\_mutex** ()
- void **std::\_\_once\_proxy** ()
- void **std::\_\_set\_once\_functor\_lock\_ptr** (unique\_lock< mutex > \*)
- 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 &, \_L2 &, \_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**
- const adopt\_lock\_t **std::adopt\_lock**
- const defer\_lock\_t **std::defer\_lock**
- const try\_to\_lock\_t **std::try\_to\_lock**

### 6.212.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).

## 6.213 nested\_exception.h File Reference

### Classes

- class [std::nested\\_exception](#)  
*Exception class with exception\_ptr data member.*

### Namespaces

- namespace [std](#)

### Defines

- #define [\\_GLIBCXX\\_NESTED\\_EXCEPTION\\_H](#)

### Functions

- `template<typename _Ex >`  
`const nested_exception * std::\_\_get\_nested\_exception (const _Ex &__ex)`
- `template<typename _Ex >`  
`void std::\_\_throw\_with\_nested (_Ex &&,...) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void std::\_\_throw\_with\_nested (_Ex &&, const nested_exception *!=0) __-`  
`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)`

#### 6.213.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [nested\\_exception.h](#).

## 6.214 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 delete](#) (void \*, void \*) throw ()
- void [operator delete](#) (void \*, const std::nothrow\_t &) throw ()
- void [operator delete](#) (void \*) throw ()
- void [operator delete\[\]](#) (void \*, void \*) throw ()
- void [operator delete\[\]](#) (void \*, const std::nothrow\_t &) throw ()
- void [operator delete\[\]](#) (void \*) throw ()
- void \* [operator new](#) (std::size\_t, void \*\_\_p) throw ()
- void \* [operator new](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void \* [operator new](#) (std::size\_t) throw (std::bad\_alloc)
- void \* [operator new\[\]](#) (std::size\_t, void \*\_\_p) throw ()
- void \* [operator new\[\]](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void \* [operator new\[\]](#) (std::size\_t) throw (std::bad\_alloc)

### Variables

- const nothrow\_t [std::nothrow](#)

### 6.214.1 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](#).

### 6.214.2 Function Documentation

#### 6.214.2.1 `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`.

#### 6.214.2.2 `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.

**6.214.2.3 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.

**6.214.2.4 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.

**6.214.2.5 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.



**6.214.2.6 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.

**6.214.2.7 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`.

**6.214.2.8 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.

**6.214.2.9 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.

**6.214.2.10 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`.

**6.214.2.11 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.

**6.214.2.12 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.

## 6.215 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](#)

### Defines

- #define [\\_NEW\\_ALLOCATOR\\_H](#)

### 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> &)

#### 6.215.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new\\_allocator.h](#).

## 6.216 numeric File Reference

### Defines

- `#define _GLIBCXX_NUMERIC`

### 6.216.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

## 6.217 numeric File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_EXT_NUMERIC`

### Functions

- `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 _ForwardIter, typename _Tp >`  
`void \_\_gnu\_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `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)`

#### 6.217.1 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](#).

## 6.218 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](#)

### Defines

- `#define GLIBCXX\_PARALLEL\_NUMERIC\_H`

### Functions

- `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 \_IIter, typename \_Tp, typename \_BinaryOperation, typename \_IteratorTag >  
\_Tp std::__parallel::__accumulate_switch (\_IIter \_\_begin, \_IIter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, \_IteratorTag)`
- `template<typename \_IIter, typename \_Tp, typename \_IteratorTag >  
\_Tp std::__parallel::__accumulate_switch (\_IIter \_\_begin, \_IIter \_\_end, \_Tp \_\_init, \_IteratorTag)`
- `template<typename \_IIter, typename \_OutputIterator, typename \_BinaryOperation >  
\_OutputIterator std::__parallel::__adjacent_difference_switch (\_IIter \_\_begin, \_IIter \_\_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 \_IIter, typename \_OutputIterator, typename \_BinaryOperation, typename \_IteratorTag1, typename \_IteratorTag2 >  
\_OutputIterator std::__parallel::__adjacent_difference_switch (\_IIter \_\_begin, \_IIter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, \_IteratorTag1, \_IteratorTag2)`
- `template<typename \_IIter1, typename \_IIter2, typename \_Tp, typename \_BinaryFunction1, typename \_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 _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 _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 _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 _Tp, typename _BinaryOperation >`
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
 - `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 >`
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init)`
 - `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, typename _BinaryOperation >`
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu_parallel::sequential_tag)`
 - `template<typename _Iter, typename _Tp >`
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, 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)`
 - `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 >`
`_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 _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 >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter _`
`__end, _OutputIterator __result, __gnu_parallel::sequential_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 >`
`_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, 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, typename _BinaryFunction1, type-`
`name _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, typename _BinaryFunction1, type-`
`name _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 >`
`_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, typename _BinaryOperation >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`

6.218.1 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](#).

6.219 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)
- #define [_EXT_NUMERIC_TRAITS](#)

6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric_traits.h](#).

6.220 numericfwd.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Defines

- #define [_GLIBCXX_PARALLEL_NUMERICFWD_H](#)

Functions

- `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 _Tp, typename _BinaryOper, typename _Tag >
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper,
_Tag)`
- `template<typename _Iter, typename _Tp, typename _Tag >
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `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 _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename
_Tag2 >
_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-
BinaryOper, _Tag1, _Tag2)`
- `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 _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 _Iter, typename _OIter, typename _BinaryOper >
_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _-
BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter std::__parallel::partial_sum_switch (_IIter, _IIter, _OIter, _-`
`BinaryOper, _Tag1, _Tag2)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_IIter, _IIter, _Tp, _BinaryOper, __gnu_`
`parallel::Parallelism)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_IIter, _IIter, _Tp, _BinaryOper, __gnu_`
`parallel::sequential_tag)`
- `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 >`
`_Tp std::__parallel::accumulate (_IIter __begin, _IIter __end, _Tp __init, __`
`gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _Tp >`
`_Tp std::__parallel::accumulate (_IIter __begin, _IIter __end, _Tp __init)`
- `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, __gnu_`
`parallel::Parallelism)`
- `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::sequential_tag)`
- `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)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename BinaryFunction1, type-`
`name BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_IIter1, _IIter1, _IIter2, _Tp, BinaryFunc-`
`tion1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, type-`
`name _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 __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, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Tp >
  _Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __-
first2, _Tp __init)
• 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)

```

6.220.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

6.221 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](#)

Defines

- #define `_GLIBCXX_PARALLEL_OMP_LOOP_H`

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)`

6.221.1 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](#).

6.222 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](#)

Defines

- #define `_GLIBCXX_PARALLEL_OMP_LOOP_STATIC_H`

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)`

6.222.1 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](#).

6.223 `os_defines.h` File Reference

Defines

- `#define __NO_CTYPE`
- `#define _GLIBCXX_OS_DEFINES`

6.223.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [os_defines.h](#).

6.224 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`

Defines

- `#define _GLIBCXX_OSTREAM`

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<class _Traits>`
`basic_ostream<char, _Traits> & std::operator<< (basic_ostream<char, _Traits> &__out, const unsigned 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 char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, const char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, const _CharT *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, unsigned 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, char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, _CharT __c)`

6.224.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

6.225 ostream.tcc File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _OSTREAM_TCC`

Functions

- template ostream & **std::endl** (ostream &)
- template ostream & **std::ends** (ostream &)
- template ostream & **std::flush** (ostream &)
- template wostream & **std::operator**<< (wostream &, const wchar_t *)
- template wostream & **std::operator**<< (wostream &, wchar_t)
- template ostream & **std::operator**<< (ostream &, const signed char *)
- template ostream & **std::operator**<< (ostream &, const unsigned char *)
- template ostream & **std::operator**<< (ostream &, const char *)
- template ostream & **std::operator**<< (ostream &, signed char)
- template ostream & **std::operator**<< (ostream &, unsigned char)
- template ostream & **std::operator**<< (ostream &, char)

- template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _-
CharT, _Traits > &__out, const char *__s)

6.225.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream.tcc](#).

6.226 ostream_insert.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _OSTREAM_INSERT_H`

Functions

- `template<typename _CharT, typename _Traits >`
`void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, stream-`
`size __n)`
- `template wostream & std::__ostream_insert (wostream &, const wchar_t *,`
`streamsize)`
- `template ostream & std::__ostream_insert (ostream &, const char *, stream-`
`size)`
- `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)`

6.226.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream_insert.h](#).

6.227 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](#)

Defines

- #define `_GLIBCXX_PARALLEL_PAR_LOOP_H`

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)`

6.227.1 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](#).

6.228 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.

Defines

- #define `_GLIBCXX_PARALLEL_PARALLEL_H`

6.228.1 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](#).

6.229 `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`

Defines

- `#define _GLIBCXX_PARALLEL_PARTIAL_SUM_H`

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)`

6.229.1 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`.

6.230 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_PARALLEL_PARTITION_H`
- `#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)`

6.230.1 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](#).

6.230.2 Define Documentation

6.230.2.1 `#define _GLIBCXX_VOLATILE`

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by [__gnu_parallel::__parallel_partition\(\)](#).

6.231 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](#)

Defines

- `#define _POD_CHAR_TRAITS_H`

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)`

6.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pod_char_traits.h](#).

6.232 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)`
- `#define _POINTER_H`

Functions

- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __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!= (_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, const`
`_Pointer_adapter< _Tp2 > &__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 _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 _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 _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 _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _`
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__`
`rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __`
`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==(_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 _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 _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 _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 _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)`

6.232.1 Detailed Description

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](#).

6.233 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](#)

Defines

- `#define _POOL_ALLOCATOR_H`

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 > &)`

6.233.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool_allocator.h](#).

6.234 postypes.h File Reference

Classes

- class [std::fpos<_StateT>](#)
Class representing stream positions.

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_POSTYPES_H`

Typedefs

- typedef 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)`

6.234.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [postypes.h](#).

6.235 `priority_queue.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.235.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority_queue.hpp](#).

6.236 `priority_queue_base_dispatch.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.236.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority_queue_base_dispatch.hpp](#).

6.237 profiler.h File Reference

Interface of the profiling runtime library.

Classes

- struct [__gnu_profile::__reentrance_guard](#)
Reentrance guard.

Namespaces

- namespace [__gnu_profile](#)

Defines

- #define [__profctx_hashtable_construct](#)(__x...)
- #define [__profctx_hashtable_construct2](#)(__x...)
- #define [__profctx_hashtable_destruct](#)(__x...)
- #define [__profctx_hashtable_destruct2](#)(__x...)
- #define [__profctx_hashtable_resize](#)(__x...)
- #define [__profctx_is_invalid](#)()
- #define [__profctx_is_off](#)()
- #define [__profctx_is_on](#)()
- #define [__profctx_list_construct](#)(__x...)
- #define [__profctx_list_construct2](#)(__x...)
- #define [__profctx_list_destruct](#)(__x...)
- #define [__profctx_list_destruct2](#)(__x...)
- #define [__profctx_list_insert](#)(__x...)
- #define [__profctx_list_invalid_operator](#)(__x...)
- #define [__profctx_list_iterate](#)(__x...)
- #define [__profctx_list_operation](#)(__x...)
- #define [__profctx_list_rewind](#)(__x...)
- #define [__profctx_map_to_unordered_map_construct](#)(__x...)
- #define [__profctx_map_to_unordered_map_destruct](#)(__x...)
- #define [__profctx_map_to_unordered_map_erase](#)(__x...)
- #define [__profctx_map_to_unordered_map_find](#)(__x...)
- #define [__profctx_map_to_unordered_map_insert](#)(__x...)
- #define [__profctx_map_to_unordered_map_invalidate](#)(__x...)
- #define [__profctx_map_to_unordered_map_iterate](#)(__x...)
- #define [__profctx_report](#)()

- #define **__profctx_turn_off()**
- #define **__profctx_turn_on()**
- #define **__profctx_vector_construct(__x...)**
- #define **__profctx_vector_construct2(__x...)**
- #define **__profctx_vector_destruct(__x...)**
- #define **__profctx_vector_destruct2(__x...)**
- #define **__profctx_vector_find(__x...)**
- #define **__profctx_vector_insert(__x...)**
- #define **__profctx_vector_invalid_operator(__x...)**
- #define **__profctx_vector_iterate(__x...)**
- #define **__profctx_vector_resize(__x...)**
- #define **__profctx_vector_resize2(__x...)**
- #define **_GLIBCXX_PROFILE_DATA(__name)**
- #define **_GLIBCXX_PROFILE_DEFINE_DATA(__type, __name, __initial_value...)**
- #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_PROFILER_H**
- #define **_GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)**
- #define **_GLIBCXX_PROFILE_THREADS**
- #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 *, size_t, size_t, size_t)**
- void **__gnu_profile::__trace_hashtable_size_construct (const void *, size_t)**
- void **__gnu_profile::__trace_hashtable_size_destruct (const void *, size_t, size_t)**
- void **__gnu_profile::__trace_hashtable_size_resize (const void *, size_t, 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 *, size_t)
- void `__gnu_profile::__trace_list_to_set_insert` (const void *, size_t, size_t)
- void `__gnu_profile::__trace_list_to_set_invalid_operator` (const void *)
- void `__gnu_profile::__trace_list_to_set_iterate` (const void *, 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 *, size_t, size_t)
- void `__gnu_profile::__trace_list_to_vector_invalid_operator` (const void *)
- void `__gnu_profile::__trace_list_to_vector_iterate` (const void *, size_t)
- void `__gnu_profile::__trace_list_to_vector_resize` (const void *, size_t, 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 *, size_t, size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void *, size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void *, size_t, size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void *, size_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void *, size_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void *, size_t, size_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void *, size_t, 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 *, size_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void *, size_t, size_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void *)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void *, size_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void *, size_t, size_t)

- `bool __gnu_profile::__turn_off ()`
- `bool __gnu_profile::__turn_on ()`

6.237.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

6.238 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](#)

Defines

- #define [_GLIBCXX_PROFILE_PROFILER_HASHTABLE_SIZE_H](#)

Functions

- void [__gnu_profile::__trace_hashtable_size_construct](#) (const void *, size_t)
- void [__gnu_profile::__trace_hashtable_size_destruct](#) (const void *, size_t, 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 *, size_t, size_t)

6.238.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler_hashtable_size.h](#).

6.239 profiler_list_to_slist.h File Reference

Diagnostics for list to slist.

Namespaces

- namespace [__gnu_profile](#)

Defines

- #define `_GLIBCXX_PROFILE_PROFILER_LIST_TO_SLIST_H`

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 *)

6.239.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler_list_to_slist.h](#).

6.240 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](#)

Defines

- #define [_GLIBCXX_PROFILE_PROFILER_LIST_TO_VECTOR_H](#)

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 *, size_t, size_t)
- void [__gnu_profile::__trace_list_to_vector_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_iterate](#) (const void *, 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 *, size_t, size_t)

6.240.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler_list_to_vector.h](#).

6.241 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](#)

Defines

- #define `_GLIBCXX_PROFILE_PROFILER_MAP_TO_UNORDERED_MAP_H`

Functions

- int [__gnu_profile::__log2](#) (size_t __size)
- float [__gnu_profile::__map_erase_cost](#) (size_t __size)
- float [__gnu_profile::__map_find_cost](#) (size_t __size)
- float [__gnu_profile::__map_insert_cost](#) (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 *, size_t, size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_find](#) (const void *, size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_init](#) ()

- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void *, size_t, size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void *, size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_report` (FILE *__f, _-_warning_vector_t &__warnings)

6.241.1 Detailed Description

Diagnostics for `map` to `unordered_map`.

Definition in file [profiler_map_to_unordered_map.h](#).

6.242 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`

Defines

- `#define _GLIBCXX_PROFILE_PROFILER_NODE_H`

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 ()`
- `__gnu_profile::__size (const __stack_t &__stack)`
- `size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__write (FILE * __f, const __stack_t __stack)`

6.242.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler_node.h](#).

6.243 profiler_state.h File Reference

Global profiler state.

Namespaces

- namespace [__gnu_profile](#)

Defines

- #define `_GLIBCXX_PROFILE_PROFILER_STATE_H`

Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

Functions

- `__state_type & __gnu_profile::__get__state ()`
- `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 ()`

6.243.1 Detailed Description

Global profiler state.

Definition in file [profiler_state.h](#).

6.244 profiler_trace.h File Reference

Diagnostics for container sizes.

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_MUTEX_INITIALIZER](#)
- #define [_GLIBCXX_IMPL_UNORDERED_MAP](#)
- #define [_GLIBCXX_PROFILE_PROFILER_TRACE_H](#)

Typedefs

- typedef `std::std::vector< __cost_factor * >` [__gnu_profile::__cost_factor_vector](#)
- typedef `pthread_mutex_t` [__gnu_profile::__mutex_t](#)
- typedef `std::std::vector< __warning_data >` [__gnu_profile::__warning_vector_t](#)

Functions

- `size_t` [__gnu_profile::__env_to_size_t](#) (const char * __env_var, size_t __default_value)
- `__cost_factor_vector * &` [__gnu_profile::__get__cost_factors](#) ()
- `__mutex_t &` [__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 ()`
- `size_t & __gnu_profile::__get__S_max_mem ()`
- `size_t & __gnu_profile::__get__S_max_stack_depth ()`
- `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 ()`
- `void __gnu_profile::__lock (__mutex_t &__m)`
- `int __gnu_profile::__log_magnitude (float f)`
- `size_t __gnu_profile::__max (size_t __a, size_t __b)`
- `size_t __gnu_profile::__max_mem ()`
- `size_t __gnu_profile::__min (size_t __a, size_t __b)`
- `FILE * __gnu_profile::__open_output_file (const char *extension)`
- `bool __gnu_profile::__profctx_init (void)`
- `void __gnu_profile::__profctx_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 ()`
- `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::__unlock` (`__mutex_t` &`__m`)
- void `__gnu_profile::__write_cost_factors` ()

6.244.1 Detailed Description

Diagnostics for container sizes. Data structures to represent profiling traces.

Definition in file [profiler_trace.h](#).

6.245 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](#)

Defines

- #define [_GLIBCXX_PROFILE_PROFILER_VECTOR_SIZE_H](#)

Functions

- void [__gnu_profile::__trace_vector_size_construct](#) (const void *, size_t)
- void [__gnu_profile::__trace_vector_size_destruct](#) (const void *, size_t, 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 *, size_t, size_t)

6.245.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler_vector_size.h](#).

6.246 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](#)

Defines

- `#define _GLIBCXX_PROFILE_PROFILER_VECTOR_TO_LIST_H`

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 *, size_t)
- void [__gnu_profile::__trace_vector_to_list_init](#) ()
- void [__gnu_profile::__trace_vector_to_list_insert](#) (const void *, size_t, size_t)
- void [__gnu_profile::__trace_vector_to_list_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_vector_to_list_iterate](#) (const void *, size_t)
- void [__gnu_profile::__trace_vector_to_list_report](#) (FILE *, [__warning_vector_t](#) &)
- void [__gnu_profile::__trace_vector_to_list_resize](#) (const void *, size_t, size_t)

6.246.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler_vector_to_list.h](#).

6.247 queue File Reference

Defines

- `#define _GLIBCXX_QUEUE`

6.247.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

6.248 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_PARALLEL_QUEUE_H](#)
- [#define _GLIBCXX_VOLATILE](#)

6.248.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

6.248.2 Define Documentation

6.248.2.1 [#define _GLIBCXX_VOLATILE](#)

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file [queue.h](#).

6.249 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](#)

Defines

- #define `_GLIBCXX_PARALLEL_QUICKSORT_H`

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)`

6.249.1 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](#).

6.250 random File Reference

Defines

- `#define _GLIBCXX_RANDOM`

6.250.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

6.251 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>`
- 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, 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 _RealType , size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `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 , 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::geometric_distribution< _IntType > &)`
- `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::cauchy_distribution< _RealType > &)`
- `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 _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 _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_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::exponential_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 _CharT, typename _Traits >`
`std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > & __is, std::bernoulli_distribution & __x)`
- `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 _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 > &)`

6.251.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.h](#).

6.252 random.tcc File Reference

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `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 _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 _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 _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, 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 > &__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 > &__os, const student_t_distribution< _RealType > &__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 > &, const std::cauchy_distribution< _RealType > &)`
- `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 _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, 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 > &, const std::exponential_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 _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 _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 _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 &)`
- `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 > &)`

- `ostream< _CharT, _Traits > &`, const `std::uniform_int_distribution< _IntType > &`)
- `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 _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 _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 _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 _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 _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 > &__is, piecewise_constant_distribution< _RealType > &__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 _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_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 > &__is, gamma_distribution< _RealType > &__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, fisher_f_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 > &)`
- `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, lognormal_distribution< _RealType > &__x)`
- `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::exponential_distribution< _RealType > &)`
- `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 _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, negative_binomial_distribution< _IntType > &__x)`
- `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 _IntType, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<_CharT, _Traits > &, std::uniform_int_distribution<_IntType > &)`
- `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 _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 _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 _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 _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)`

6.252.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.tcc](#).

6.253 `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](#)

Defines

- `#define _GLIBCXX_PARALLEL_RANDOM_NUMBER_H`

6.253.1 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](#).

6.254 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`

Defines

- `#define _GLIBCXX_PARALLEL_RANDOM_SHUFFLE_H`

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)`

6.254.1 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](#).

6.255 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_greater< _R1, _R2 >`
ratio_greater
- struct `std::ratio_greater_equal< _R1, _R2 >`
ratio_greater_equal
- struct `std::ratio_less< _R1, _R2 >`
ratio_less
- struct `std::ratio_less_equal< _R1, _R2 >`
ratio_less_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`

Defines

- `#define _GLIBCXX_RATIO`

Typedefs

- `typedef ratio< 1, 1000000000000000000 > std::atto`
- `typedef ratio< 1, 100 > std::centi`
- `typedef ratio< 10, 1 > std::deca`
- `typedef ratio< 1, 10 > std::deci`
- `typedef ratio< 1000000000000000000, 1 > std::exa`
- `typedef ratio< 1, 1000000000000000 > std::femto`
- `typedef ratio< 1000000000, 1 > std::giga`
- `typedef ratio< 100, 1 > std::hecto`
- `typedef ratio< 1000, 1 > std::kilo`
- `typedef ratio< 1000000, 1 > std::mega`
- `typedef ratio< 1, 1000000 > std::micro`
- `typedef ratio< 1, 1000 > std::milli`
- `typedef ratio< 1, 1000000000 > std::nano`
- `typedef ratio< 1000000000000000000, 1 > std::peta`
- `typedef ratio< 1, 1000000000000 > std::pico`
- `typedef ratio< 1000000000000, 1 > std::tera`

6.255.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

6.256 rb_tree File Reference

Classes

- struct [__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)

Defines

- #define [_RB_TREE](#)

6.256.1 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](#).

6.257 rc_string_base.h File Reference

Classes

- class [__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)

Defines

- `#define _RC_STRING_BASE_H`

6.257.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [rc_string_base.h](#).

6.258 regex File Reference

Defines

- `#define _GLIBCXX_REGEX`

6.258.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

6.259 regex File Reference

The common implementation file for `tr1` and `std` 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_error`
*A regular expression *exception* class.
 The regular expression library throws objects of this class on error.*
- 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`
- namespace `std::regex_constants`

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::wsub_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, class _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!=](#) (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 _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 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 _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<](#) (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, 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, 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 _BiIter >`
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__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 >`
`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 _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 _BiIter >`
`bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__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==(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 _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 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 _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>(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, 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, 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 _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>= (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, 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, 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 _BiIter >`
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__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)`

Matching, Searching, and Replacing

- `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 _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 _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, 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 _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 _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >
 >
 bool [std::regex_match](#) (_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 _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)
 - 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 _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 _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_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_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 _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 _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)

```

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::format_default](#)
- static const match_flag_type [std::regex_constants::format_first_only](#)
- static const match_flag_type [std::regex_constants::format_no_copy](#)
- static const match_flag_type [std::regex_constants::format_sed](#)
- static const match_flag_type [std::regex_constants::match_any](#)
- static const match_flag_type [std::regex_constants::match_continuous](#)
- 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_bow](#)
- static const match_flag_type [std::regex_constants::match_not_eol](#)
- static const match_flag_type [std::regex_constants::match_not_eow](#)
- static const match_flag_type [std::regex_constants::match_not_null](#)
- static const match_flag_type [std::regex_constants::match_prev_avail](#)

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::awk`
- static const `syntax_option_type` `std::regex_constants::basic`
- 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::egrep`
- static const `syntax_option_type` `std::regex_constants::extended`
- static const `syntax_option_type` `std::regex_constants::grep`
- 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`

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_backref` (`_S_error_backref`)
- static const `error_type` `std::regex_constants::error_badbrace` (`_S_error_-`
`badbrace`)
- static const `error_type` `std::regex_constants::error_badrepeat` (`_S_error_-`
`badrepeat`)
- static const `error_type` `std::regex_constants::error_brace` (`_S_error_brace`)
- static const `error_type` `std::regex_constants::error_brack` (`_S_error_brack`)
- static const `error_type` `std::regex_constants::error_collate` (`_S_error_collate`)
- static const `error_type` `std::regex_constants::error_complexity` (`_S_error_-`
`complexity`)
- 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_paren` (`_S_error_paren`)
- 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_stack` (`_S_error_stack`)

6.259.1 Detailed Description

The common implementation file for `tr1` and `std` regular expressions. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/regex](#).

6.260 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)`
- `#define _ROPE`

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)`
- `void __gnu_cxx::_S_cond_store_eos (wchar_t &__c)`
- `void __gnu_cxx::_S_cond_store_eos (char &__c)`
- `template<class _CharT >`
`void __gnu_cxx::_S_cond_store_eos (_CharT &)`
- `template<class _CharT >`
`_CharT __gnu_cxx::_S_eos (_CharT *)`
- `bool __gnu_cxx::_S_is_basic_char_type (wchar_t *)`
- `bool __gnu_cxx::_S_is_basic_char_type (char *)`
- `template<class _CharT >`
`bool __gnu_cxx::_S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::_S_is_one_byte_char_type (char *)`
- `template<class _CharT >`
`bool __gnu_cxx::_S_is_one_byte_char_type (_CharT *)`
- `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 > &__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 >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, _CharT __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, const rope< _CharT, _Alloc > &__right)`
- `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_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_-`
`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_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _-`
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, _CharT __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, const rope< _CharT, _Alloc > &__right)`
- `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_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_const_iterator< _CharT, _Alloc`
`> &__x, const _Rope_const_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< _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 _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_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< _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< _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 _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_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 > &__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< _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 >`
`void __gnu_cxx::swap(rope< _CharT, _Alloc > &__x, 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)`

Variables

- `rope<_CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn<_CharT, _Alloc >)`

6.260.1 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](#).

6.261 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)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `template<class _CharT >`
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `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)`

6.261.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ropeimpl.h](#).

6.262 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](#)

Defines

- `#define _GLIBCXX_DEBUG_SAFE_BASE_H`

6.262.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_base.h](#).

6.263 safe_iterator.h File Reference

Classes

- class [__gnu_debug::Safe_iterator<_Iterator, _Sequence>](#)
Safe iterator wrapper.

Namespaces

- namespace [__gnu_debug](#)

Defines

- #define [_GLIBCXX_DEBUG_SAFE_ITERATOR_H](#)

Functions

- bool [__gnu_debug::__check_singular_aux](#) (const [_Safe_iterator_base](#) * __x)
- 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](#) >
[_Safe_iterator<_Iterator, _Sequence>](#) [__gnu_debug::operator+](#) (typename [_Safe_iterator<_Iterator, _Sequence>](#)::[difference_type](#) __n, const [_Safe_iterator<_Iterator, _Sequence>](#) & __i)
- template<typename [_Iterator](#) , typename [_Sequence](#) >
[_Safe_iterator<_Iterator, _Sequence>](#)::[difference_type](#) [__gnu_debug::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](#) [__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 _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 _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)`

6.263.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.h](#).

6.264 safe_iterator.tcc File Reference

Namespaces

- namespace [__gnu_debug](#)

Defines

- #define `_GLIBCXX_DEBUG_SAFE_ITERATOR_TCC`

6.264.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.tcc](#).

6.265 safe_sequence.h File Reference

Classes

- class `__gnu_debug::_After_nth_from<_Iterator >`
- 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`

Defines

- `#define _GLIBCXX_DEBUG_SAFE_SEQUENCE_H`

6.265.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_sequence.h](#).

6.266 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](#)

Defines

- `#define _GLIBCXX_PARALLEL_SEARCH_H`

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)`

6.266.1 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](#).

6.267 set File Reference

Defines

- `#define _GLIBCXX_SET`

6.267.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

6.268 set File Reference

Defines

- `#define _GLIBCXX_DEBUG_SET`

6.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

6.269 set File Reference

Defines

- `#define _GLIBCXX_PROFILE_SET`

6.269.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

6.270 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](#)

Defines

- #define [_GLIBCXX_DEBUG_SET_H](#)

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)`

6.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

6.271 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](#)

Defines

- `#define _GLIBCXX_PROFILE_SET_H`

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)`

6.271.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

6.272 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](#)

Defines

- #define `_GLIBCXX_PARALLEL_SET_OPERATIONS_H`

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)`

6.272.1 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](#).

6.273 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\(__c\)`
- `#define _GLIBCXX_PARALLEL_SETTINGS_H`

6.273.1 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.

6.273.2 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](#).

6.273.3 Define Documentation

6.273.3.1 `#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](#).

6.274 `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::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`

Defines

- `#define _SHARED_PTR_H`

Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr<_Tp>` `std::allocate_shared` (`_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 _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `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 _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 (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

6.274.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr.h](#).

6.275 shared_ptr_base.h File Reference

Defines

- #define `_SHARED_PTR_BASE_H`

6.275.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr_base.h](#).

6.276 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)`
- `#define _SLICE_ARRAY_H`

6.276.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [slice_array.h](#).

6.277 slist File Reference

Classes

- class [__gnu_cxx::slist< _Tp, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Defines

- #define [_SLIST](#)

Functions

- [_Slist_node_base * __gnu_cxx::__slist_make_link](#) ([_Slist_node_base * __prev_node](#), [_Slist_node_base * __new_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_previous](#) ([_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 * __head](#))
- [void __gnu_cxx::__slist_splice_after](#) ([_Slist_node_base * __pos](#), [_Slist_node_base * __before_first](#), [_Slist_node_base * __before_last](#))
- [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)`

6.277.1 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](#).

6.278 `sort.h` File Reference

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace `__gnu_parallel`

Defines

- `#define _GLIBCXX_PARALLEL_SORT_H`

Functions

- `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)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, _Parallelism __parallelism)`

6.278.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

6.279 sso_string_base.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Defines

- #define [_SSO_STRING_BASE_H](#)

6.279.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sso_string_base.h](#).

6.280 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_iostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Namespaces

- namespace `std`

Defines

- `#define _GLIBCXX_SSTREAM`

6.280.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [sstream](#).

6.281 sstream.tcc File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _SSTREAM_TCC`

6.281.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sstream.tcc](#).

6.282 stack File Reference

Defines

- `#define _GLIBCXX_STACK`

6.282.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stack](#).

6.283 `standard_policies.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

Enumerations

- enum { `default_store_hash` }

6.283.1 Detailed Description

Contains standard policies for containers.

Definition in file [standard_policies.hpp](#).

6.284 stdatomic.h File Reference

Defines

- #define `_GLIBCXX_STDATOMIC_H`

6.284.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdatomic.h](#).

6.285 `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](#)

Defines

- `#define _GLIBCXX_STDEXCEPT`

6.285.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdexcept](#).

6.286 `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`

Defines

- `#define _STDIO_FILEBUF_H`

6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_filebuf.h](#).

6.287 `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`

Defines

- `#define _STDIO_SYNC_FILEBUF_H`

6.287.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_sync_filebuf.h](#).

6.288 stl_algo.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STL_ALGO_H`

Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }

Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::__copy_n (_RandomAccessIterator __first, _Size __n, _-`
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::__copy_n (_InputIterator __first, _Size __n, _-`
`OutputIterator __result, input_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 _BidirectionalIterator1 , typename _BidirectionalIterator2 >`
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_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 _ForwardIterator1 , typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag)`
- `template<typename _RandomAccessIterator , typename _Predicate >`
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_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_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 , typename _Compare >`
`void std::heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _ForwardIterator , typename _Predicate , typename _Distance >`
`_ForwardIterator std::inplace_stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__inplace_stable_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__inplace_stable_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `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, typename _Compare >`
`void std::__introsort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __-`
`depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __nth, _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 _RandomAccessIterator, typename _Size >`
`void std::__introsort_loop (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Size __depth_limit)`
- `long long std::__lg (long long __n)`
- `long std::__lg (long __n)`
- `int std::__lg (int __n)`
- `template<typename _Size >`
`_Size std::__lg (_Size __n)`
- `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 _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 _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 _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _-  
BidirectionalIterator3 >`  

```

_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-
first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-
BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)

```
  - `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 _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename  
_Distance >`  

```

void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-
RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance
__step_size)

```
  - `template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >`  

```

void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)

```
  - `template<typename _RandomAccessIterator , typename _Pointer >`  

```

void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Pointer __buffer)

```
  - `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 _BidirectionalIterator , typename _Distance >`  

```

void std::__merge_without_buffer (_BidirectionalIterator __first, _-
BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance
__len1, _Distance __len2)

```
  - `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 _BidirectionalIterator , typename _Predicate >`  

```

_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _-
BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)

```
  - `template<typename _ForwardIterator , typename _Predicate >`  

```

_FowardIterator std::__partition (_ForwardIterator __first, _ForwardIterator _-
last, _Predicate __pred, forward_iterator_tag)

```

- `template<typename _RandomAccessIterator >`  
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`__last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`  
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last, forward_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 _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 _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 >`  
`_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 >`  
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator _-`  
`__last, _Integer __count, const _Tp &__val, std::forward_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 , typename`  
`_Compare >`

```

void std::stable_sort_adaptive (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,
_Compare __comp)
• 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 _Compare >
void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Compare __comp)
• template<typename _RandomAccessIterator >
void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _-
RandomAccessIterator __last)
• template<typename _RandomAccessIterator, typename _Compare >
void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare
__comp)
• template<typename _RandomAccessIterator >
void std::__unguarded_linear_insert (_RandomAccessIterator __last)
• template<typename _RandomAccessIterator, typename _Tp, typename _Compare >
_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator _-
__first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)
• template<typename _RandomAccessIterator, typename _Tp >
_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator _-
__first, _RandomAccessIterator __last, const _Tp &__pivot)
• template<typename _RandomAccessIterator, typename _Compare >
_RandomAccessIterator std::__unguarded_partition_pivot (_-
RandomAccessIterator __first, _RandomAccessIterator __last, _Compare
__comp)
• template<typename _RandomAccessIterator >
_RandomAccessIterator std::__unguarded_partition_pivot (_-
RandomAccessIterator __first, _RandomAccessIterator __last)
• template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >
_ForumIterator std::unique_copy (_InputIterator __first, _InputIterator _-
__last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_-
iterator_tag, forward_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 _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 _ForwardIterator >
_ForumIterator std::unique_copy (_InputIterator __first, _InputIterator _-
__last, _ForwardIterator __result, input_iterator_tag, forward_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 _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `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, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val)`
- `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, 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 _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, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __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 _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, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `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 _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `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 _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `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 >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `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, typename _Compare >`  
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __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 _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`

- `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, 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 _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, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`  
`__predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `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)`
- `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, 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, type-
name _Compare >`
`_OutputIterator std::set_union (_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 _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, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
 - `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
 - `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, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `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)`

6.288.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_algo.h](#).

6.289 `stl_algobase.h` File Reference

Namespaces

- namespace `std`

Defines

- `#define _GLIBCXX_MOVE3(Tp, Up, Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(Tp, Up, Vp)`
- `#define _STL_ALGOBASE_H`

Functions

- `template<bool_IsMove, typename _II, typename _OI >`
`_OI std::copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool_IsMove, typename _II, typename _OI >`
`_OI std::copy_move_a2 (_II __first, _II __last, _OI __result)`
- `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 _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, ostreambuf_`
`iterator< _CharT, char_traits< _CharT >`
`> >::type std::copy_move_a2`
`(_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT >`
`>)`
- `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 _Tp >`
`__gnu_cxx::enable_if< __is_byte< _Tp >::value, void >::type std::`
`fill_a (_Tp * __first, _Tp * __last, const _Tp & __c)`

- `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 _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 _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 _II1, typename _II2 >`
`bool std::lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 _-`
`__first2, _II2 __last2)`
- `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 _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`
`BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `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 _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)`

6.289.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_algobase.h](#).

6.290 `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`

Defines

- `#define _STL_BVECTOR_H`

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_const_iterator` `std::operator+` (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `_Bit_iterator` `std::operator+` (`ptrdiff_t __n`, `const _Bit_iterator &__x`)
- `ptrdiff_t` `std::operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)

6.290.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std::bvector.h](#).

6.291 `stl_construct.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STL_CONSTRUCT_H`

Functions

- `template<typename _T1 , typename _T2 >`
`void std::_Construct (_T1 * __p, _T2 && __value)`
- `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)`

6.291.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_construct.h](#).

6.292 `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`
- `#define _STL_DEQUE_H`

Functions

- `size_t std::__deque_buf_size (size_t __size)`
- `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 (_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 >`
`_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, _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 (_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 >`
`_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, 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 _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `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 _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 _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 _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 _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 >`
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

6.292.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_deque.h](#).

6.292.2 Define Documentation

6.292.2.1 `#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 82 of file `stl_deque.h`.

6.293 `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`

Defines

- `#define _STL_FUNCTION_H`

Functions

- `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 > mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)())`
- `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 _Arg1, typename _Arg2, typename _Result > pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))`
- `template<typename _Arg, typename _Result > pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_-Arg))`

6.293.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_function.h](#).

6.294 `std_heap.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STL_HEAP_H`

Functions

- `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, typename _Tp >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `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, typename _Compare >`
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)`

- `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, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `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)`

6.294.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_heap.h](#).

6.295 `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)`
- `#define _STL_ITERATOR_H`

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 move_iterator<_IteratorL> &__x, const move_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 _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 >`
`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 > __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 _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `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, 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 _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.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 >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_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 _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 _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_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 _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 _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_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 _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 _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_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 _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 _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_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 _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)`

6.295.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

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 [stl_iterator.h](#).

6.296 `stl_iterator_base_funcs.h` File Reference

Namespaces

- namespace `std`

Defines

- `#define _STL_ITERATOR_BASE_FUNCS_H`

Functions

- `template<typename _RandomAccessIterator, typename _Distance >`
`void std::advance (_RandomAccessIterator &__i, _Distance __n, random_`
`access_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void std::advance (_BidirectionalIterator &__i, _Distance __n,`
`bidirectional_iterator_tag)`
- `template<typename _InputIterator, typename _Distance >`
`void std::advance (_InputIterator &__i, _Distance __n, input_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)`

6.296.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility functions, such as [distance\(\)](#) and [advance\(\)](#).

Definition in file [stl_iterator_base_funcs.h](#).

6.297 `std_iterator_base_types.h` File Reference

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::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
Common iterator class.
- struct `std::iterator_traits< _Iterator >`
Traits class for iterators.
- 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`

Defines

- `#define _STL_ITERATOR_BASE_TYPES_H`

Functions

- `template<typename _Iter >`
`iterator_traits< _Iter >::iterator_category` [std::__iterator_category](#) (const `_Iter`
&)

6.297.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file [stl_iterator_base_types.h](#).

6.298 `std_list.h` File Reference

Classes

- 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.
- struct `std::_List_node_base`
Common part of a 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`

Defines

- `#define _STL_LIST_H`

Functions

- `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 _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 >`
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

6.298.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_list.h](#).

6.299 `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`

Defines

- `#define _STL_MAP_H`

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)`

6.299.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_map.h](#).

6.300 `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`

Defines

- `#define _STL_MULTIMAP_H`

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)`

6.300.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_multimap.h](#).

6.301 `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`

Defines

- `#define _STL_MULTISSET_H`

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)`

6.301.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_multiset.h](#).

6.302 `std_numeric.h` File Reference

Namespaces

- namespace `std`

Defines

- `#define _STL_NUMERIC_H`

Functions

- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation > _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _Tp > _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `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)`

6.302.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_numeric.h](#).

6.303 `std_pair.h` File Reference

Classes

- struct `std::pair< _T1, _T2 >`
pair holds two objects of arbitrary type.

Namespaces

- namespace `std`

Defines

- `#define _STL_PAIR_H`

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 >`
`bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`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)`

6.303.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_pair.h](#).

6.304 `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`

Defines

- `#define _STL_QUEUE_H`

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)`

6.304.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_queue.h](#).

6.305 `stl_raw_storage_iter.h` File Reference

Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp>`

Namespaces

- namespace `std`

Defines

- `#define _STL_RAW_STORAGE_ITERATOR_H`

6.305.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `stl_raw_storage_iter.h`.

6.306 stl_relops.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::rel_ops](#)

Defines

- `#define _STL_RELOPS_H`

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)`

6.306.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

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

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file [stl_relops.h](#).

6.307 `std_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`

Defines

- `#define _STL_SET_H`

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)`

6.307.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_set.h](#).

6.308 `std::stack.h` File Reference

Classes

- class `std::stack<_Tp, _Sequence >`
A standard container giving FILO behavior.

Namespaces

- namespace `std`

Defines

- `#define _STL_STACK_H`

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)`

6.308.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_stack.h](#).

6.309 `stl_tempbuf.h` File Reference

Classes

- class `std::_Temporary_buffer<_ForwardIterator, _Tp >`

Namespaces

- namespace `std`

Defines

- `#define _STL_TEMPBUF_H`

Functions

- `template<typename _Tp >`
`pair<_Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _Tp >`
`void std::return_temporary_buffer (_Tp *__p)`

6.309.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_tempbuf.h](#).

6.310 `stl_tree.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STL_TREE_H`

Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

Functions

- `std::_attribute__` (`(__pure__)`) `bool __verify_grouping(const char *__grouping`
- 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)`
- `size_t const string &__grouping_tmp std::throw ()`

6.310.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_tree.h](#).

6.311 `std_uninitialized.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STL_UNINITIALIZED_H`

Functions

- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_construct_range (_ForwardIterator __first, _-`
`ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _-`
`InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _-`
`InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)`
- `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 _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_RandomAccessIterator __-`
`first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size _-`
`_n, _ForwardIterator __result, input_iterator_tag)`
- `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 _Tp2 >`
`void std::uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`
`_Tp &__x, allocator< _Tp2 > &)`
- `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 _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)`

6.311.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_uninitialized.h](#).

6.312 `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`

Defines

- `#define _STL_VECTOR_H`

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)`

6.312.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_vector.h](#).

6.313 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`

Defines

- `#define _STREAM_ITERATOR_H`

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)`

6.313.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stream_iterator.h](#).

6.314 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](#)

Defines

- `#define _GLIBXX_STREAMBUF`

Functions

- `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)
- `template<typename _CharT, typename _Traits >`
streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)

6.314.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

6.315 streambuf.tcc File Reference

Namespaces

- namespace [std](#)

Defines

- #define `_STREAMBUF_TCC`

Functions

- template streamsize **std::__copy_streambufs** (basic_streambuf< wchar_t > *, basic_streambuf< wchar_t > *)
- template streamsize **std::__copy_streambufs** (basic_streambuf< char > *, basic_streambuf< char > *)
- template<typename _CharT, typename _Traits > streamsize **std::__copy_streambufs** (basic_streambuf< _CharT, _Traits > * __sbin, basic_streambuf< _CharT, _Traits > * __sbout)
- 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)
- template<typename _CharT, typename _Traits > streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)

6.315.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf.tcc](#).

6.316 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`

Defines

- `#define _STREAMBUF_ITERATOR_H`

Functions

- `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<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, ostreambuf_iterator< _CharT > >::__type`
`std::__copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _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)`

6.316.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf_iterator.h](#).

6.317 string File Reference

Defines

- `#define _GLIBCXX_STRING`

6.317.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

6.318 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`

Defines

- `#define _GLIBCXX_DEBUG_STRING`

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)`
- 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`>
`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 >`
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const`
`basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const`
`basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (-`
`CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const`
`_CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > __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 _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 >`
`std::basic_ostream< _CharT, _Traits > & __gnu_debug::operator<<`
`(std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT,`
`_Traits, _Allocator > &__str)`
- `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 >`
`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 >`
`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 >`
`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 >`
`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)`

6.318.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

6.319 stringfwd.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _STRINGFWD_H`

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`

6.319.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stringfwd.h](#).

6.320 `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`

Defines

- `#define _GLIBCXX_SYSTEM_ERROR`

Functions

- `_GLIBCXX_CONST` `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_condition &__lhs, const error_condition &__rhs)`

- bool **std::operator!=** (const error_condition &__lhs, const error_code &__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_condition &__rhs)
- bool **std::operator<** (const error_code &__lhs, const error_code &__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_condition &__lhs, const error_code &__rhs)

- bool **std::operator==** (const error_code &__lhs, const error_condition &__rhs)

- bool **std::operator==** (const error_code &__lhs, const error_code &__rhs)
- `_GLIBCXX_CONST` const error_category & **std::system_category** () throw ()

Variables

- error_code **std::make_error_code** (errc)
- error_condition **std::make_error_condition** (errc)

6.320.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system_error](#).

6.321 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`

6.321.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag_and_trait.hpp](#).

6.322 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](#)

Defines

- `#define _GLIBCXX_PARALLEL_TAGS_H`

6.322.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

6.323 tgm $.h$ File Reference

Defines

- #define `_GLIBCXX_TGMATH_H`

6.323.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgm \$.h\$](#) .

6.324 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`

Defines

- `#define _GLIBCXX_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 ()`

6.324.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

6.325 `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`

Defines

- `#define _THROW_ALLOCATOR_H`

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)`

6.325.1 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](#).

6.326 `time_members.h` File Reference

Namespaces

- namespace [std](#)

6.326.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [time_members.h](#).

6.327 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`

6.327.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_policy.hpp](#).

6.328 `tree_trace_base.hpp` File Reference

6.328.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_trace_base.hpp](#).

6.329 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**

6.329.1 Detailed Description

Contains trie-related policies.

Definition in file [trie_policy.hpp](#).

6.330 tuple File Reference

Classes

- struct `std::_Tuple_impl<_Idx >`
- struct `std::_Tuple_impl<_Idx, _Head, _Tail...>`
- class `std::tuple<_Elements >`
tuple
- 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`

Defines

- `#define _GLIBCXX_TUPLE`

Functions

- `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<std::size_t __i, typename _Head, typename... _Tail>`
`__add_ref<_Head >::type std::__get_helper (_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...> &, 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...> &, const tuple< _UElements...> &_u, const __index_holder< _UIdx...> &)`
- `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 (tuple< _TElements...>`
`&&__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`
`TElements...> &__t, const tuple< _UElements...> &__u)`

6.330.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

6.331 `type_traits` File Reference

Classes

- struct `std::__declval_protector< _Tp >`
declval
- struct `std::add_lvalue_reference< _Tp >`
add_lvalue_reference
- struct `std::add_rvalue_reference< _Tp >`
add_rvalue_reference
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
conditional
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
enable_if
- struct `std::has_nothrow_assign< _Tp >`
has_nothrow_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_assign< _Tp >`
has_trivial_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::is_base_of< _Base, _Derived >`
is_base_of
- struct `std::is_constructible< _Tp, _Args >`
is_constructible
- struct `std::is_convertible< _From, _To >`
is_convertible
- struct `std::is_explicitly_convertible< _From, _To >`
is_explicitly_convertible
- struct `std::is_lvalue_reference< typename >`
is_lvalue_reference
- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- 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_unsigned< _Tp >`
is_unsigned
- struct `std::make_signed< _Tp >`
make_signed

- struct `std::make_unsigned<_Tp>`
make_unsigned
- struct `std::remove_reference<_Tp>`
remove_reference

Namespaces

- namespace `std`

Defines

- `#define _GLIBCXX_TYPE_TRAITS`

Functions

- `template<typename _Tp>`
`add_rvalue_reference<_Tp>::type` **std::declval** ()

6.331.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type_traits](#).

6.332 type_traits File Reference

Classes

- 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_pointer< _Tp >`
add_pointer
- struct `std::add_volatile< _Tp >`
add_volatile
- struct `std::alignment_of< _Tp >`
alignment_of
- struct `std::extent< typename, _Uint >`
extent
- 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_class< _Tp >`
is_class

- struct `std::is_compound< _Tp >`
is_compound
- struct `std::is_const< typename >`
is_const
- struct `std::is_empty< _Tp >`
is_empty
- struct `std::is_enum< _Tp >`
is_enum
- 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_member_function_pointer< _Tp >`
is_member_function_pointer
- struct `std::is_member_object_pointer< _Tp >`
is_member_object_pointer
- struct `std::is_object< _Tp >`
is_object
- struct `std::is_pointer< _Tp >`
is_pointer
- struct `std::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::is_same< typename, typename >`
is_same

- struct `std::is_scalar< _Tp >`
is_scalar
- struct `std::is_union< _Tp >`
is_union
- struct `std::is_void< _Tp >`
is_void
- struct `std::is_volatile< typename >`
is_volatile
- 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_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`

Typedefs

- `typedef integral_constant< bool, false > std::false_type`
- `typedef integral_constant< bool, true > std::true_type`

6.332.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/type_traits](#).

6.333 type_traits.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Defines

- #define `_EXT_TYPE_TRAITS`

Functions

- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type)`
- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`

6.333.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type_traits.h](#).

6.334 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**

6.334.1 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](#).

6.335 typeid 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`

6.335.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeid](#).

6.336 `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)`
- `#define _TYPELIST_H`

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)`

6.336.1 Detailed Description

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

6.337 types.h File Reference

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- `#define _GLIBCXX_PARALLEL_TYPES_H`

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](#)

6.337.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

6.338 `types_traits.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.338.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types_traits.hpp](#).

6.339 `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](#)

Defines

- `#define _GLIBCXX_PARALLEL_UNIQUE_COPY_H`

Functions

- `template<typename _Iter, class _OutputIterator >`
`_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`
`_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

6.339.1 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](#).

6.340 `unique_ptr.h` File Reference

Classes

- struct `std::default_delete<_Tp>`
Primary template, [default_delete](#).
- struct `std::default_delete<_Tp[]>`
Specialization, [default_delete](#).
- class `std::unique_ptr<_Tp, _Tp_Deleter>`
20.7.12.2 [unique_ptr](#) for single objects.
- class `std::unique_ptr<_Tp[], _Tp_Deleter>`
20.7.12.3 [unique_ptr](#) for array objects with a runtime length

Namespaces

- namespace `std`

Defines

- `#define _UNIQUE_PTR_H`

Functions

- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter>`
`bool std::operator!= (const unique_ptr<_Tp, _Tp_Deleter> &__x, const unique_ptr<_Up, _Up_Deleter> &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter>`
`bool std::operator< (const unique_ptr<_Tp, _Tp_Deleter> &__x, const unique_ptr<_Up, _Up_Deleter> &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter>`
`bool std::operator<= (const unique_ptr<_Tp, _Tp_Deleter> &__x, const unique_ptr<_Up, _Up_Deleter> &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter>`
`bool std::operator== (const unique_ptr<_Tp, _Tp_Deleter> &__x, const unique_ptr<_Up, _Up_Deleter> &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter>`
`bool std::operator> (const unique_ptr<_Tp, _Tp_Deleter> &__x, const unique_ptr<_Up, _Up_Deleter> &__y)`

- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter >`
`void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _`
`Tp_Deleter > &__y)`

6.340.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unique_ptr.h](#).

6.341 unordered_map File Reference

Defines

- #define `_GLIBCXX_UNORDERED_MAP`

6.341.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_map](#).

6.342 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`

Defines

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

Functions

- `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)`

6.342.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_map](#).

6.343 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_PROFILE_UNORDERED_MAP`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `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)`

6.343.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_map](#).

6.344 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>`
`void std::swap(unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &_x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &_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, 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)`

6.344.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `unordered_map.h`.

6.345 unordered_set File Reference

Defines

- #define `_GLIBCXX_UNORDERED_SET`

6.345.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_set](#).

6.346 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`

Defines

- `#define _GLIBCXX_DEBUG_UNORDERED_SET`

Functions

- `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)`

6.346.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_set](#).

6.347 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_PROFILE_UNORDERED_SET`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `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)`

6.347.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_set](#).

6.348 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 >`
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__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, 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)`

6.348.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered_set.h](#).

6.349 utility File Reference

Defines

- `#define _GLIBCXX_UTILITY`

6.349.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

6.350 utility File Reference

Namespaces

- namespace [std](#)

Functions

- `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)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >
tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (std::pair< _Tp1, _Tp2 > &__in)`

6.350.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/utility](#).

6.351 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)`
- #define `_GLIBCXX_VALARRAY`

Functions

- `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, _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, _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, _Constant, _ValArray, _Tp, _Tp>, typename`
`__fun<__modulus, _Tp>::result_type > std::operator% (const _Tp &__t,`
`const valarray<_Tp> &__v)`

- `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, _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< __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, _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, _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<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, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _-`
`Tp > &__v, const _Tp &__t)`
- `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, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `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< __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< __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, _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< __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< __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, _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< __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, _Constant, _Tp, _Tp >, typename _-`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const _Tp &__t)`

- `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< __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, _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, _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< __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< __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, _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< __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, _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, _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_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< __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< __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, _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, _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, _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, _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)`
- `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< __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)`
- `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)`

6.351.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

6.352 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)
- #define **_VALARRAY_AFTER_H**

Functions

- `template<typename _Tp >
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `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< _Acos, _ValArray, _Tp >, _Tp > std::acos (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< _Asin, _ValArray, _Tp >, _Tp > std::asin (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< _Atan, _ValArray, _Tp >, _Tp > std::atan (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< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`

- `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, _ValArray, _ValArray, _Tp, _Tp >, _Tp >`
`std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `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<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, type`
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename`
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `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 _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, type`
`name _Dom1::value_type > std::atan2 (const _Expr< _Dom1, typename _`
`Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_`
`type > &__e2)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (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< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (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< _Exp, _ValArray, _Tp >, _Tp > std::exp (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< _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<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`
`std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `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 _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, _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, _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 _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< __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< __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, _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, _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 >, typename __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< __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 _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, _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, _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 _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __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 _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, _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, _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, _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< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __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< __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< __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, _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, _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 _Dom1 , class _Dom2 >
 _Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __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< __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 _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, _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< __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 _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< __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 _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)

- 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, _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, _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< __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< __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 _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, _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, _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 _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< __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 _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, _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, _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 _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< __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 _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, _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, _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 _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< __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< __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, _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 _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< __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, _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, _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 _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 _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< __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 _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, _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, _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 _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_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 _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, _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< __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< __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< __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, _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, _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 _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< __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 _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 _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 _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __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_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< __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 >`

- 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, _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 _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< __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, _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 _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 _Dom1, class _Dom2 >

_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __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<typename _Tp >

_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > **std::pow** (const _Tp &__t, const valarray< _Tp > &__v)
 - template<typename _Tp >

_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > **std::pow** (const valarray< _Tp > &__v, const _Tp &__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, _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<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<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, _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< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (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< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (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< _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<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`
`std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`

6.352.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_after.h](#).

6.353 valarray_array.h File Reference

Namespaces

- namespace [std](#)

Defines

- #define `_DEFINE_ARRAY_FUNCTION(_Op, _Name)`
- #define `_VALARRAY_ARRAY_H`

Functions

- `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 (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `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, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `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 (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `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, _Tp *__restrict __b, const size_t *__restrict __i)`
- `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 (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, _Tp *__restrict __b, size_t __n, size_t __s)`
- `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, size_t __n, _Tp *__restrict __b)`
- `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 (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `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 (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 (_Array< _Tp > __a, _Array< size_t > __i, 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, size_t __n, 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 (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`

- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict__ __a, 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< 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< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `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, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `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_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< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _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, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`n, const _Tp &__t)`
- `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,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`bool > __m, _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, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `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, size_t __s,`
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _-`
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _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,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`

- `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 >`
`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, _Array< bool > __m, _Array< _Tp > __b, 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, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `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, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _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, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `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 >`
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `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, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `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< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`size_t __s, _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, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `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, _-`
`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, 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, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`

- `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 >`
`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)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `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< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`
`> __m, _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, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `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, size_t __s,`
`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 >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`size_t __s, _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, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `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 >`
`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, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `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, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, 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, _Array< _`
`Tp > __b, size_t __n, size_t __s)`
- `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 >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`

- `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___plus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `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, _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, _Array< size_t >`
`__i, _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, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `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, 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)`
- `template<typename _Tp >`
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `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 >`
`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, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`

- `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< size_t > __i)`
- `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, 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< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::_Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `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 >`
`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, size_t __n, const _Tp &__t)`
- `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, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `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 >`
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`

- `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, _Array<`
`_Tp > __b, size_t __n, size_t __s)`
- `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, 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)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`

6.353.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_array.h](#).

6.354 `valarray_array.tcc` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _VALARRAY_ARRAY_TCC`

Functions

- `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 >`
`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< 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, class _Dom >`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a)`
- `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 > __a, size_t __n, _Array< _Tp >`
`__b, _Array< bool > __m)`
- `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_construct (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `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_fill (_Array< _Tp > __a, size_t __n, _Array< bool >`
`__m, const _Tp &__t)`

6.354.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_array.tcc](#).

6.355 `valarray_before.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _VALARRAY_BEFORE_H`

6.355.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_before.h](#).

6.356 vector File Reference

Defines

- `#define _GLIBCXX_VECTOR`

6.356.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

6.357 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`

Defines

- `#define _GLIBCXX_DEBUG_VECTOR`

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)`

6.357.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

6.358 vector File Reference

Classes

- struct `std::hash< __profile::vector< bool, _Alloc > >`
std::hash specialization for vector<bool>.

Namespaces

- namespace `std`
- namespace `std::__profile`

Defines

- `#define _GLIBCXX_PROFILE_VECTOR`

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)`

6.358.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).

6.359 `vector.tcc` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _VECTOR_TCC`

6.359.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vector.tcc](#).

6.360 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`

Defines

- `#define _VSTRING_H`

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)`
- `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>`
`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>`
`__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+ (_-`
`_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, 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 _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, 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 _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, 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, 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 __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 _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, 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, 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 double __val)`
- `__vstring __gnu_cxx::to_string (double __val)`
- `__vstring __gnu_cxx::to_string (float __val)`
- `__vstring __gnu_cxx::to_string (unsigned long long __val)`
- `__vstring __gnu_cxx::to_string (long long __val)`
- `__vstring __gnu_cxx::to_string (unsigned long __val)`
- `__vstring __gnu_cxx::to_string (long __val)`
- `__vstring __gnu_cxx::to_string (unsigned __val)`
- `__vstring __gnu_cxx::to_string (int __val)`
- `__wvstring __gnu_cxx::to_wstring (long double __val)`
- `__wvstring __gnu_cxx::to_wstring (double __val)`
- `__wvstring __gnu_cxx::to_wstring (float __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned long long __val)`

- `__wvstring __gnu_cxx::to_wstring` (long long __val)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long __val)
- `__wvstring __gnu_cxx::to_wstring` (long __val)
- `__wvstring __gnu_cxx::to_wstring` (unsigned __val)
- `__wvstring __gnu_cxx::to_wstring` (int __val)

6.360.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

6.361 `vstring.tcc` File Reference

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Defines

- `#define _VSTRING_TCC`

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+ (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+ (_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, 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 > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`

```
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,  
_Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base  
> &__str)
```

6.361.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring.tcc](#).

6.362 `vstring_fwd.h` File Reference

Namespaces

- namespace `__gnu_cxx`

Defines

- `#define _VSTRING_FWD_H`

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`

6.362.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring_fwd.h](#).

6.363 `vstring_util.h` File Reference

Namespaces

- namespace `__gnu_cxx`

Defines

- `#define _VSTRING_UTIL_H`

6.363.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring_util.h](#).

6.364 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`
- `#define _GLIBCXX_PARALLEL_WORKSTEALING_H`

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`)

6.364.1 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, 1214
- ~_RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1257
- ~_Safe_sequence_base
 - __gnu_debug::_Safe_sequence_base, 1065
- ~__versa_string
 - __gnu_cxx::__versa_string, 803
- ~auto_ptr
 - std::auto_ptr, 1628
- ~basic_filebuf
 - std::basic_filebuf, 1648
- ~basic_fstream
 - std::basic_fstream, 1688
- ~basic_ifstream
 - std::basic_ifstream, 1757
- ~basic_ios
 - std::basic_ios, 1811
- ~basic_iostream
 - std::basic_iostream, 1846
- ~basic_istream
 - std::basic_istream, 1914
- ~basic_istreamream
 - std::basic_istreamream, 1971
- ~basic_ofstream
 - std::basic_ofstream, 2027
- ~basic_ostream
 - std::basic_ostream, 2073
- ~basic_ostreamream
 - std::basic_ostreamream, 2121
- ~basic_regex
 - std::basic_regex, 2161
- ~basic_streambuf
 - std::basic_streambuf, 2173
- ~basic_string
 - __gnu_debug::basic_string, 1076
 - std::basic_string, 2201
- ~basic_stringstream
 - std::basic_stringstream, 2291
- ~collate
 - std::collate, 2435
- ~ctype
 - std::ctype< char >, 2477
 - std::ctype< wchar_t >, 2491
- ~deque
 - std::deque, 2550
- ~facet
 - std::locale::facet, 2859
- ~forward_list
 - std::forward_list, 2618
- ~gslice
 - numeric_arrays, 92
- ~ios_base
 - std::ios_base, 2741
- ~locale
 - std::locale, 2852
- ~match_results
 - std::match_results, 2904
- ~messages
 - std::messages, 2921
- ~money_get
 - std::money_get, 2934
- ~money_put
 - std::money_put, 2940
- ~moneypunct
 - std::moneypunct, 2948
- ~num_get
 - std::num_get, 3030
- ~num_put
 - std::num_put, 3042

- ~numpunct
 - std::numpunct, 3096
- ~sentry
 - std::basic_ostream::sentry, 2108
- ~stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 970
- ~temporary_buffer
 - __gnu_cxx::temporary_buffer, 1025
- ~time_get
 - std::time_get, 3293
- ~time_put
 - std::time_put, 3314
- ~type_info
 - std::type_info, 3331
- ~vector
 - std::vector, 3373
- /mnt/share/src/ Directory Reference, 331
- /mnt/share/src/gcc-trunk/ Directory Reference, 320
- /mnt/share/src/gcc-trunk/libstdc++-v3/ Directory Reference, 325
- /mnt/share/src/gcc-trunk/libstdc++-v3/doc/ Directory Reference, 316
- /mnt/share/src/gcc-trunk/libstdc++-v3/doc/doxygen/ Directory Reference, 317
- /mnt/share/src/gcc-trunk/libstdc++-v3/libsupc++/ Directory Reference, 326
- __gnu_parallel
 - parallel_balanced, 401
 - parallel_omp_loop, 401
 - parallel_omp_loop_static, 401
 - parallel_taskqueue, 402
 - parallel_unbalanced, 401
 - sequential, 401
- _AlgorithmStrategy
 - __gnu_parallel, 401
- _BALLOC_ALIGN_BYTES
 - bitmap_allocator.h, 3480
- _BinIndex
 - __gnu_parallel, 400
- _Bit_scan_forward
 - __gnu_cxx, 358
- _CASable
 - __gnu_parallel, 400
- _CASable_bits
 - __gnu_parallel, 447
- _CASable_mask
 - __gnu_parallel, 447
- _Construct
 - std, 614
- _DRandomShufflingGlobalData
 - __gnu_parallel::-_DRandomShufflingGlobalData, 1180
- _Destroy
 - std, 614, 615
- _FindAlgorithm
 - __gnu_parallel, 401
- _Find_first
 - SGIextensions, 17
- _Find_next
 - SGIextensions, 17
- _GLIBCXX_ASSERTIONS
 - compiletime_settings.h, 3537
- _GLIBCXX_ATOMIC_PROPERTY
 - atomics, 254
- _GLIBCXX_BAL_QUICKSORT
 - features.h, 3618
- _GLIBCXX_CALL
 - compiletime_settings.h, 3537
- _GLIBCXX_DEBUG_VERIFY
 - macros.h, 3709
- _GLIBCXX_DEQUE_BUF_SIZE
 - stl_deque.h, 3898
- _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
 - features.h, 3618
- _GLIBCXX_FIND_EQUAL_SPLIT
 - features.h, 3618
- _GLIBCXX_FIND_GROWING_BLOCKS
 - features.h, 3619
- _GLIBCXX_MERGESORT
 - features.h, 3619
- _GLIBCXX_PARALLEL_CONDITION
 - settings.h, 3857
- _GLIBCXX_PARALLEL_LENGTH
 - multiway_merge.h, 3744
- _GLIBCXX_QUICKSORT

- features.h, [3619](#)
- `_GLIBCXX_RANDOM_SHUFFLE_-`
 - `CONSIDER_L1`
 - compiletime_settings.h, [3538](#)
- `_GLIBCXX_RANDOM_SHUFFLE_-`
 - `CONSIDER_TLB`
 - compiletime_settings.h, [3538](#)
- `_GLIBCXX_SCALE_DOWN_FPU`
 - compiletime_settings.h, [3538](#)
- `_GLIBCXX_TREE_DYNAMIC_-`
 - `BALANCING`
 - features.h, [3619](#)
- `_GLIBCXX_TREE_FULL_COPY`
 - features.h, [3620](#)
- `_GLIBCXX_TREE_INITIAL_-`
 - `SPLITTING`
 - features.h, [3620](#)
- `_GLIBCXX_VERBOSE_LEVEL`
 - compiletime_settings.h, [3538](#)
- `_GLIBCXX_VOLATILE`
 - partition.h, [3777](#)
 - queue.h, [3805](#)
- `_GuardedIterator`
 - `__gnu_parallel::_GuardedIterator`, [1189](#)
- `_LoserTreeBase`
 - `__gnu_parallel::_LoserTreeBase`, [1214](#)
- `_M_allocate_and_copy`
 - std::vector, [3373](#)
- `_M_allocate_single_object`
 - `__gnu_cxx::bitmap_allocator`, [877](#)
- `_M_attach`
 - `__gnu_debug::_Safe_iterator`, [1047](#)
 - `__gnu_debug::_Safe_iterator_base`, [1057](#)
- `_M_attach_single`
 - `__gnu_debug::_Safe_iterator`, [1047](#)
 - `__gnu_debug::_Safe_iterator_base`, [1057](#)
- `_M_attached_to`
 - `__gnu_debug::_Safe_iterator`, [1047](#)
 - `__gnu_debug::_Safe_iterator_base`, [1057](#)
- `_M_begin`
 - `__gnu_parallel::_Piece`, [1242](#)
- `_M_bin_proc`
 - `__gnu_parallel::_`
 - `DRandomShufflingGlobalData`, [1180](#)
- `_M_bins_begin`
 - `__gnu_parallel::_DRSSorterPU`, [1182](#)
- `_M_buf`
 - `__gnu_cxx::enc_filebuf`, [908](#)
 - `__gnu_cxx::stdio_filebuf`, [992](#)
 - std::basic_filebuf, [1669](#)
- `_M_buf_locale`
 - `__gnu_cxx::enc_filebuf`, [908](#)
 - `__gnu_cxx::stdio_filebuf`, [992](#)
 - `__gnu_cxx::stdio_sync_filebuf`, [1019](#)
 - std::basic_filebuf, [1669](#)
 - std::basic_streambuf, [2190](#)
 - std::basic_stringbuf, [2275](#)
- `_M_buf_size`
 - `__gnu_cxx::enc_filebuf`, [908](#)
 - `__gnu_cxx::stdio_filebuf`, [992](#)
 - std::basic_filebuf, [1669](#)
- `_M_clear`
 - `__gnu_cxx::free_list`, [918](#)
- `_M_comp`
 - `__gnu_parallel::_LoserTree`, [1207](#)
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, [1211](#)
 - `__gnu_parallel::_LoserTreeBase`, [1215](#)
- `_M_const_iterators`
 - `__gnu_debug::_Safe_sequence`, [1063](#)
 - `__gnu_debug::_Safe_sequence_base`, [1066](#)
 - `__gnu_debug::basic_string`, [1123](#)
 - std::__debug::bitset, [1410](#)
 - std::__debug::deque, [1416](#)
 - std::__debug::list, [1423](#)
 - std::__debug::map, [1428](#)
 - std::__debug::multimap, [1434](#)
 - std::__debug::multiset, [1440](#)
 - std::__debug::set, [1446](#)
 - std::__debug::unordered_map, [1452](#)

- std::__debug::unordered_multimap, 1457
- std::__debug::unordered_multiset, 1461
- std::__debug::unordered_set, 1467
- std::__debug::vector, 1473
- _M_create_node
 - std::list, 2831
- _M_create_pback
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 971
 - std::basic_filebuf, 1648
- _M_data
 - std::_List_node, 1551
- _M_deallocate_single_object
 - __gnu_cxx::bitmap_allocator, 877
- _M_dereferenceable
 - __gnu_debug::_Safe_iterator, 1048
- _M_destroy_pback
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 971
 - std::basic_filebuf, 1648
- _M_detach
 - __gnu_debug::_Safe_iterator, 1048
 - __gnu_debug::_Safe_iterator_base, 1057
- _M_detach_all
 - __gnu_debug::_Safe_sequence, 1061
 - __gnu_debug::_Safe_sequence_base, 1065
 - __gnu_debug::basic_string, 1077
 - std::__debug::bitset, 1409
 - std::__debug::deque, 1415
 - std::__debug::list, 1421
 - std::__debug::map, 1426
 - std::__debug::multimap, 1432
 - std::__debug::multiset, 1438
 - std::__debug::set, 1444
 - std::__debug::unordered_map, 1450
 - std::__debug::unordered_multimap, 1455
 - std::__debug::unordered_multiset, 1460
 - std::__debug::unordered_set, 1465
 - std::__debug::vector, 1471
- _M_dist
 - __gnu_parallel::_-DRandomShufflingGlobalData, 1180
- _M_elements_leftover
 - __gnu_parallel::_QSBThreadLocal, 1252
- _M_end
 - __gnu_parallel::_Piece, 1242
- _M_ext_buf
 - __gnu_cxx::enc_filebuf, 908
 - __gnu_cxx::stdio_filebuf, 992
 - std::basic_filebuf, 1669
- _M_ext_buf_size
 - __gnu_cxx::enc_filebuf, 908
 - __gnu_cxx::stdio_filebuf, 992
 - std::basic_filebuf, 1669
- _M_ext_next
 - __gnu_cxx::enc_filebuf, 908
 - __gnu_cxx::stdio_filebuf, 993
 - std::basic_filebuf, 1670
- _M_fill_initialize

-
- std::deque, 2550
 - _M_finish_iterator
 - __gnu_parallel::__accumulate_-selector, 1127
 - __gnu_parallel::__adjacent_difference_selector, 1128
 - __gnu_parallel::__count_if_selector, 1137
 - __gnu_parallel::__count_selector, 1139
 - __gnu_parallel::__fill_selector, 1141
 - __gnu_parallel::__for_each_-selector, 1147
 - __gnu_parallel::__generate_-selector, 1149
 - __gnu_parallel::__generic_for_each_selector, 1152
 - __gnu_parallel::__identity_selector, 1154
 - __gnu_parallel::__inner_product_-selector, 1157
 - __gnu_parallel::__replace_if_-selector, 1169
 - __gnu_parallel::__replace_selector, 1172
 - __gnu_parallel::__transform1_-selector, 1174
 - __gnu_parallel::__transform2_-selector, 1176
 - _M_first
 - __gnu_parallel::__Job, 1197
 - _M_first_insert
 - __gnu_parallel::__LoserTree, 1207
 - __gnu_parallel::__LoserTree< false, _Tp, _Compare >, 1211
 - __gnu_parallel::__LoserTreeBase, 1215
 - _M_gcount
 - std::basic_fstream, 1738
 - std::basic_ifstream, 1794
 - std::basic_iostream, 1895
 - std::basic_istream, 1950
 - std::basic_istreamstream, 2008
 - std::basic_stringstream, 2340
 - _M_get
 - __gnu_cxx::free_list, 918
 - _M_get_distance
 - __gnu_debug::_Safe_iterator, 1048
 - _M_get_mutex
 - __gnu_debug::_Safe_iterator, 1048
 - __gnu_debug::_Safe_iterator_base, 1057
 - __gnu_debug::_Safe_sequence, 1061
 - __gnu_debug::_Safe_sequence_-base, 1065
 - __gnu_debug::basic_string, 1077
 - std::__debug::bitset, 1409
 - std::__debug::deque, 1415
 - std::__debug::list, 1421
 - std::__debug::map, 1427
 - std::__debug::multimap, 1433
 - std::__debug::multiset, 1439
 - std::__debug::set, 1445
 - std::__debug::unordered_map, 1450
 - std::__debug::unordered_multimap, 1455
 - std::__debug::unordered_multiset, 1460
 - std::__debug::unordered_set, 1465
 - std::__debug::vector, 1471
 - _M_get_result
 - std::__basic_future, 1385
 - std::future, 2650
 - std::future< _Res & >, 2653
 - std::future< void >, 2656
 - std::shared_future, 3246
 - std::shared_future< _Res & >, 3250
 - std::shared_future< void >, 3253
 - _M_getloc
 - std::basic_fstream, 1689
 - std::basic_ifstream, 1758
 - std::basic_ios, 1811
 - std::basic_iostream, 1846
 - std::basic_istream, 1914
 - std::basic_istreamstream, 1972
 - std::basic_ofstream, 2027
 - std::basic_ostream, 2073
 - std::basic_ostreamstream, 2121
 - std::basic_stringstream, 2292
 - std::ios_base, 2741
 - _M_global
-

- __gnu_parallel::_QSBThreadLocal, 1252
- _M_in_beg
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 993
 - __gnu_cxx::stdio_sync_filebuf, 1019
 - std::basic_filebuf, 1670
 - std::basic_streambuf, 2191
 - std::basic_stringbuf, 2275
- _M_in_cur
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 993
 - __gnu_cxx::stdio_sync_filebuf, 1019
 - std::basic_filebuf, 1670
 - std::basic_streambuf, 2191
 - std::basic_stringbuf, 2275
- _M_in_end
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 994
 - __gnu_cxx::stdio_sync_filebuf, 1020
 - std::basic_filebuf, 1671
 - std::basic_streambuf, 2191
 - std::basic_stringbuf, 2276
- _M_incrementable
 - __gnu_debug::_Safe_iterator, 1049
- _M_initial
 - __gnu_parallel::_QSBThreadLocal, 1252
- _M_initialize_map
 - std::_Deque_base, 1525
 - std::deque, 2551
- _M_insert
 - __gnu_cxx::free_list, 919
- _M_invalidate
 - __gnu_debug::_Safe_iterator, 1049
- _M_invalidate_all
 - __gnu_debug::_Safe_sequence, 1061
 - __gnu_debug::_Safe_sequence_base, 1066
 - __gnu_debug::basic_string, 1077
 - std::_debug::bitset, 1409
 - std::_debug::deque, 1415
 - std::_debug::list, 1421
 - std::_debug::map, 1427
 - std::_debug::multimap, 1433
 - std::_debug::multiset, 1439
 - std::_debug::set, 1445
 - std::_debug::unordered_map, 1450
 - std::_debug::unordered_multimap, 1455
 - std::_debug::unordered_multiset, 1460
 - std::_debug::unordered_set, 1465
 - std::_debug::vector, 1471
- _M_invalidate_if
 - __gnu_debug::_Safe_sequence, 1062
 - __gnu_debug::basic_string, 1077
 - std::_debug::deque, 1416
 - std::_debug::list, 1422
 - std::_debug::map, 1427
 - std::_debug::multimap, 1433
 - std::_debug::multiset, 1439
 - std::_debug::set, 1445
 - std::_debug::unordered_map, 1451
 - std::_debug::unordered_multimap, 1456
 - std::_debug::unordered_multiset, 1461
 - std::_debug::unordered_set, 1466
 - std::_debug::vector, 1472
- _M_invalidate_single
 - __gnu_debug::_Safe_iterator, 1049
- _M_is_begin
 - __gnu_debug::_Safe_iterator, 1049
- _M_is_end
 - __gnu_debug::_Safe_iterator, 1050
- _M_iterators
 - __gnu_debug::_Safe_sequence, 1063
 - __gnu_debug::_Safe_sequence_base, 1066
 - __gnu_debug::basic_string, 1123
 - std::_debug::bitset, 1410
 - std::_debug::deque, 1417
 - std::_debug::list, 1423
 - std::_debug::map, 1428
 - std::_debug::multimap, 1434

- std::__debug::multiset, 1440
- std::__debug::set, 1446
- std::__debug::unordered_map, 1452
- std::__debug::unordered_multimap, 1457
- std::__debug::unordered_multiset, 1462
- std::__debug::unordered_set, 1467
- std::__debug::vector, 1473
- _M_key
 - __gnu_parallel::LoserTreeBase::_Loser, 1217
- _M_last
 - __gnu_parallel::Job, 1197
- _M_leftover_parts
 - __gnu_parallel::_QSBThreadLocal, 1252
- _M_load
 - __gnu_parallel::Job, 1197
- _M_log_k
 - __gnu_parallel::LoserTree, 1207
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, 1211
 - __gnu_parallel::LoserTreeBase, 1216
- _M_losers
 - __gnu_parallel::LoserTree, 1207
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, 1212
 - __gnu_parallel::LoserTreeBase, 1216
- _M_mode
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 994
 - std::basic_filebuf, 1671
 - std::basic_stringbuf, 2276
- _M_new_elements_at_back
 - std::deque, 2551
- _M_new_elements_at_front
 - std::deque, 2552
- _M_next
 - __gnu_debug::_Safe_iterator, 1053
 - __gnu_debug::_Safe_iterator_base, 1058
- _M_num_bins
 - __gnu_parallel::_DRandomShufflingGlobalData, 1180
- _M_num_bits
 - __gnu_parallel::_DRandomShufflingGlobalData, 1180
- _M_num_threads
 - __gnu_parallel::_DRSSorterPU, 1183
 - __gnu_parallel::PMWMSortingData, 1245
 - __gnu_parallel::_QSBThreadLocal, 1253
- _M_offsets
 - __gnu_parallel::PMWMSortingData, 1245
- _M_out_beg
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 994
 - __gnu_cxx::stdio_sync_filebuf, 1020
 - std::basic_filebuf, 1671
 - std::basic_streambuf, 2191
 - std::basic_stringbuf, 2276
- _M_out_cur
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 994
 - __gnu_cxx::stdio_sync_filebuf, 1020
 - std::basic_filebuf, 1671
 - std::basic_streambuf, 2192
 - std::basic_stringbuf, 2276
- _M_out_end
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 995
 - __gnu_cxx::stdio_sync_filebuf, 1021
 - std::basic_filebuf, 1672
 - std::basic_streambuf, 2192
 - std::basic_stringbuf, 2277
- _M_pback
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 995
 - std::basic_filebuf, 1672
- _M_pback_cur_save

- __gnu_cxx::enc_filebuf, 911
- __gnu_cxx::stdio_filebuf, 995
- std::basic_filebuf, 1672
- _M_pback_end_save
 - __gnu_cxx::enc_filebuf, 911
 - __gnu_cxx::stdio_filebuf, 996
 - std::basic_filebuf, 1673
- _M_pback_init
 - __gnu_cxx::enc_filebuf, 911
 - __gnu_cxx::stdio_filebuf, 996
 - std::basic_filebuf, 1673
- _M_pieces
 - __gnu_parallel::_-
PMWMSSortingData, 1246
- _M_pop_back_aux
 - std::deque, 2552
- _M_pop_front_aux
 - std::deque, 2552
- _M_prior
 - __gnu_debug::_Safe_iterator, 1053
 - __gnu_debug::_Safe_iterator_base, 1058
- _M_push_back_aux
 - std::deque, 2552
- _M_push_front_aux
 - std::deque, 2552
- _M_range_check
 - std::deque, 2553
 - std::vector, 3373
- _M_range_initialize
 - std::deque, 2553
- _M_reading
 - __gnu_cxx::enc_filebuf, 911
 - __gnu_cxx::stdio_filebuf, 996
 - std::basic_filebuf, 1673
- _M_reallocate_map
 - std::deque, 2554
- _M_reserve_elements_at_back
 - std::deque, 2554
- _M_reserve_elements_at_front
 - std::deque, 2554
- _M_reserve_map_at_back
 - std::deque, 2554
- _M_reserve_map_at_front
 - std::deque, 2555
- _M_revalidate_singular
 - __gnu_debug::_Safe_sequence, 1062
 - __gnu_debug::_Safe_sequence_base, 1066
 - __gnu_debug::basic_string, 1078
 - std::_debug::bitset, 1410
 - std::_debug::deque, 1416
 - std::_debug::list, 1422
 - std::_debug::map, 1427
 - std::_debug::multimap, 1433
 - std::_debug::multiset, 1439
 - std::_debug::set, 1445
 - std::_debug::unordered_map, 1451
 - std::_debug::unordered_multimap, 1456
 - std::_debug::unordered_multiset, 1461
 - std::_debug::unordered_set, 1466
 - std::_debug::vector, 1472
- _M_samples
 - __gnu_parallel::_-
PMWMSSortingData, 1246
- _M_sd
 - __gnu_parallel::_DRSSorterPU, 1183
- _M_seed
 - __gnu_parallel::_DRSSorterPU, 1183
- _M_sequence
 - __gnu_debug::_Safe_iterator, 1053
 - __gnu_debug::_Safe_iterator_base, 1058
- _M_sequential_algorithm
 - __gnu_parallel::_adjacent_find_selector, 1130
 - __gnu_parallel::_find_first_of_selector, 1143
 - __gnu_parallel::_find_if_selector, 1144
 - __gnu_parallel::_mismatch_selector, 1160
- _M_set_buffer
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 971
 - std::basic_filebuf, 1649
- _M_set_node

-
- std::_Deque_iterator, 1528
 - _M_source
 - __gnu_parallel::_-
 - DRandomShufflingGlobalData, 1181
 - __gnu_parallel::_LoserTreeBase::_-
 - Loser, 1217
 - __gnu_parallel::_-
 - PMWMSSortingData, 1246
 - _M_starts
 - __gnu_parallel::_-
 - DRandomShufflingGlobalData, 1181
 - __gnu_parallel::_-
 - PMWMSSortingData, 1246
 - _M_sup
 - __gnu_parallel::_LoserTreeBase::_-
 - Loser, 1217
 - _M_swap
 - __gnu_debug::_Safe_sequence, 1062
 - __gnu_debug::_Safe_sequence_base, 1066
 - __gnu_debug::basic_string, 1078
 - std::_debug::bitset, 1410
 - std::_debug::deque, 1416
 - std::_debug::list, 1422
 - std::_debug::map, 1428
 - std::_debug::multimap, 1434
 - std::_debug::multiset, 1440
 - std::_debug::set, 1446
 - std::_debug::unordered_map, 1451
 - std::_debug::unordered_multimap, 1456
 - std::_debug::unordered_multiset, 1461
 - std::_debug::unordered_set, 1466
 - std::_debug::vector, 1472
 - _M_temporaries
 - __gnu_parallel::_-
 - DRandomShufflingGlobalData, 1181
 - _M_temporary
 - __gnu_parallel::_-
 - PMWMSSortingData, 1247
 - _M_transfer_iter
 - __gnu_debug::_Safe_sequence, 1062
 - __gnu_debug::basic_string, 1078
 - std::_debug::deque, 1416
 - std::_debug::list, 1422
 - std::_debug::map, 1428
 - std::_debug::multimap, 1434
 - std::_debug::multiset, 1440
 - std::_debug::set, 1446
 - std::_debug::unordered_map, 1451
 - std::_debug::unordered_multimap, 1456
 - std::_debug::unordered_multiset, 1461
 - std::_debug::unordered_set, 1466
 - std::_debug::vector, 1472
 - _M_use_pointer
 - __gnu_parallel::_LoserTreeTraits, 1231
 - _M_version
 - __gnu_debug::_Safe_iterator, 1054
 - __gnu_debug::_Safe_iterator_base, 1058
 - __gnu_debug::_Safe_sequence, 1063
 - __gnu_debug::_Safe_sequence_base, 1067
 - __gnu_debug::basic_string, 1124
 - std::_debug::bitset, 1411
 - std::_debug::deque, 1417
 - std::_debug::list, 1423
 - std::_debug::map, 1428
 - std::_debug::multimap, 1434
 - std::_debug::multiset, 1440
 - std::_debug::set, 1446
 - std::_debug::unordered_map, 1452
 - std::_debug::unordered_multimap, 1457
 - std::_debug::unordered_multiset, 1462
 - std::_debug::unordered_set, 1467
 - std::_debug::vector, 1473
 - _M_w
 - std::_Base_bitset, 1518
 - _M_write
 - std::basic_fstream, 1689
-

- std::basic_iostream, 1847
- std::basic_ofstream, 2027
- std::basic_ostream, 2073
- std::basic_ostringstream, 2122
- std::basic_stringstream, 2292
- _MultiwayMergeAlgorithm
 - __gnu_parallel, 401
- _Parallelism
 - __gnu_parallel, 401
- _PartialSumAlgorithm
 - __gnu_parallel, 402
- _Piece
 - __gnu_parallel::_QSBThreadLocal, 1251
- _PseudoSequence
 - __gnu_parallel::_PseudoSequence, 1248
- _QSBThreadLocal
 - __gnu_parallel::_QSBThreadLocal, 1252
- _RandomNumber
 - __gnu_parallel::_RandomNumber, 1254
- _RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1257
- _Safe_iterator
 - __gnu_debug::_Safe_iterator, 1045, 1046
- _Safe_iterator_base
 - __gnu_debug::_Safe_iterator_base, 1056
- _SequenceIndex
 - __gnu_parallel, 400
- _SortAlgorithm
 - __gnu_parallel, 402
- _SplittingAlgorithm
 - __gnu_parallel, 402
- _Temporary_buffer
 - std::_Temporary_buffer, 1578
- _ThreadIndex
 - __gnu_parallel, 401
- _Unchecked_flip
 - std::bitset, 2373
- _Unchecked_reset
 - std::bitset, 2373
- _Unchecked_set
 - SGIextensions, 17
 - std::bitset, 2373
- _Unchecked_test
 - std::bitset, 2373
- __atomic0::atomic_address, 763
- __atomic0::atomic_bool, 765
- __atomic0::atomic_flag, 766
- __atomic2::atomic_address, 767
- __atomic2::atomic_bool, 768
- __atomic2::atomic_flag, 769
- __attribute__
 - __gnu_debug::_Safe_iterator, 1046
 - __gnu_debug::_Safe_iterator_base, 1057
 - __gnu_parallel::_Settings, 1262
- __begin1_iterator
 - __gnu_parallel::_inner_product_selector, 1156
- __begin2_iterator
 - __gnu_parallel::_inner_product_selector, 1157
- __bins_end
 - __gnu_parallel::_DRSSorterPU, 1182
- __bit_allocate
 - __gnu_cxx::_detail, 371
- __bit_free
 - __gnu_cxx::_detail, 371
- __calc_borders
 - __gnu_parallel, 402
- __check_dereferenceable
 - __gnu_debug, 379
- __check_singular
 - __gnu_debug, 380
- __check_singular_aux
 - __gnu_debug, 380
- __check_string
 - __gnu_debug, 380
- __compare_and_swap
 - __gnu_parallel, 402
- __compare_and_swap_32
 - __gnu_parallel, 403
- __compare_and_swap_64
 - __gnu_parallel, 403

-
- __ctype_type
 - std::basic_fstream, 1683
 - std::basic_ifstream, 1752
 - std::basic_ios, 1806
 - std::basic_iostream, 1841
 - std::basic_istream, 1909
 - std::basic_istream, 1966
 - std::basic_ofstream, 2022
 - std::basic_ostream, 2068
 - std::basic_ostringstream, 2116
 - std::basic_stringstream, 2286
 - __cxxabiv1::__forced_unwind, 770
 - __decode2
 - __gnu_parallel, 404
 - __delete_min_insert
 - __gnu_parallel::_LoserTree, 1206
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1210
 - __determine_samples
 - __gnu_parallel, 404
 - __encode2
 - __gnu_parallel, 405
 - __fetch_and_add
 - __gnu_parallel, 405
 - __fetch_and_add_32
 - __gnu_parallel, 405
 - __fetch_and_add_64
 - __gnu_parallel, 406
 - __final_insertion_sort
 - std, 601
 - __find
 - std, 601
 - __find_if
 - std, 601, 602
 - __find_if_not
 - std, 602
 - __find_template
 - __gnu_parallel, 406–408
 - __for_each_template_random_access
 - __gnu_parallel, 409
 - __for_each_template_random_access_ed
 - __gnu_parallel, 409
 - __for_each_template_random_access_omp_loop
 - __gnu_parallel, 410
 - __for_each_template_random_access_omp_loop_static
 - __gnu_parallel, 411
 - __for_each_template_random_access_workstealing
 - __gnu_parallel, 411
 - __gcd
 - std, 602
 - __genrand_bits
 - __gnu_parallel::_RandomNumber, 1255
 - __get__global_lock
 - __gnu_profile, 456
 - __get_min_source
 - __gnu_parallel::_LoserTree, 1206
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1210
 - __gnu_parallel::_LoserTreeBase, 1215
 - __get_num_threads
 - __gnu_parallel::balanced_quick_sort_tag, 1274
 - __gnu_parallel::balanced_tag, 1276
 - __gnu_parallel::default_parallel_tag, 1279
 - __gnu_parallel::exact_tag, 1282
 - __gnu_parallel::multiway_mergesort_exact_tag, 1286
 - __gnu_parallel::multiway_mergesort_sampling_tag, 1288
 - __gnu_parallel::multiway_mergesort_tag, 1290
 - __gnu_parallel::omp_loop_static_tag, 1292
 - __gnu_parallel::omp_loop_tag, 1294
 - __gnu_parallel::parallel_tag, 1297
 - __gnu_parallel::quicksort_tag, 1299
 - __gnu_parallel::sampling_tag, 1301
 - __gnu_parallel::unbalanced_tag, 1304
 - __glibcxx_check_erase
 - macros.h, 3707
 - __glibcxx_check_erase_range
 - macros.h, 3707
-

-
- __glibcxx_check_heap_pred
 macros.h, 3708
 - __glibcxx_check_insert
 macros.h, 3708
 - __glibcxx_check_insert_range
 macros.h, 3708
 - __glibcxx_check_partitioned_lower
 macros.h, 3708
 - __glibcxx_check_partitioned_lower_pred
 macros.h, 3708
 - __glibcxx_check_partitioned_upper_pred
 macros.h, 3709
 - __glibcxx_check_sorted_pred
 macros.h, 3709
 - __gnu_cxx, 335
 - _Bit_scan_forward, 358
 - __static_pointer_cast, 357
 - operator<, 362
 - operator<=, 363, 364
 - operator>, 366, 367
 - operator>=, 367, 368
 - operator+, 359–361
 - operator==, 364, 365
 - swap, 368
 - __gnu_cxx::_Caster, 857
 - __gnu_cxx::_Char_types, 858
 - __gnu_cxx::_ExtPtr_allocator, 859
 - __gnu_cxx::_Invalid_type, 861
 - __gnu_cxx::_Pointer_adapter, 862
 - __gnu_cxx::_Relative_pointer_impl, 865
 - __gnu_cxx::_Relative_pointer_impl<
 const_Tp >, 866
 - __gnu_cxx::_Std_pointer_impl, 867
 - __gnu_cxx::_Unqualified_type, 868
 - __gnu_cxx::_common_pool_policy, 771
 - __gnu_cxx::_detail, 370
 - __bit_allocate, 371
 - __bit_free, 371
 - __num_bitmaps, 371
 - __num_blocks, 371
 - __gnu_cxx::_detail::_Bitmap_counter,
 774
 - __gnu_cxx::_detail::_Ffit_finder, 775
 - argument_type, 776
 - result_type, 776
 - __gnu_cxx::_detail::_mini_vector, 772
 - __gnu_cxx::_mt_alloc, 777
 - __gnu_cxx::_mt_alloc_base, 779
 - __gnu_cxx::_per_type_pool_policy, 781
 - __gnu_cxx::_pool< false >, 782
 - __gnu_cxx::_pool< true >, 784
 - __gnu_cxx::_pool_alloc, 786
 - __gnu_cxx::_pool_alloc_base, 788
 - __gnu_cxx::_pool_base, 790
 - __gnu_cxx::_rc_string_base, 791
 - __gnu_cxx::_scoped_lock, 794
 - __gnu_cxx::_versa_string, 795
 - ~__versa_string, 803
 - __versa_string, 800–803
 - append, 804–806
 - assign, 807–810
 - at, 810, 811
 - back, 811
 - begin, 812
 - c_str, 812
 - capacity, 812
 - cbegin, 813
 - cend, 813
 - clear, 813
 - compare, 813–817
 - copy, 817
 - crbegin, 818
 - crend, 818
 - data, 818
 - empty, 819
 - end, 819
 - erase, 819, 820
 - find, 821, 822
 - find_first_not_of, 823, 824
 - find_first_of, 825, 826
 - find_last_not_of, 827–829
 - find_last_of, 829–831
 - front, 831, 832
 - get_allocator, 832
 - insert, 832–837
 - length, 837
 - max_size, 838
 - npos, 856
 - operator+=", 838, 839
 - operator=, 839–841
 - push_back, 842
 - rbegin, 842
-

- rend, 842, 843
- replace, 843–849
- reserve, 850
- resize, 851
- rfind, 852, 853
- shrink_to_fit, 854
- size, 854
- substr, 854
- swap, 855
- __gnu_cxx::annotate_base, 869
- __gnu_cxx::array_allocator, 870
- __gnu_cxx::array_allocator_base, 872
- __gnu_cxx::binary_compose, 874
 - argument_type, 875
 - result_type, 875
- __gnu_cxx::bitmap_allocator, 876
 - _M_allocate_single_object, 877
 - _M_deallocate_single_object, 877
- __gnu_cxx::char_traits, 879
- __gnu_cxx::character, 881
- __gnu_cxx::condition_base, 882
- __gnu_cxx::constant_binary_fun, 883
- __gnu_cxx::constant_unary_fun, 884
- __gnu_cxx::constant_void_fun, 885
- __gnu_cxx::debug_allocator, 886
- __gnu_cxx::enc_filebuf, 887
 - _M_buf, 908
 - _M_buf_locale, 908
 - _M_buf_size, 908
 - _M_create_pback, 892
 - _M_destroy_pback, 892
 - _M_ext_buf, 908
 - _M_ext_buf_size, 908
 - _M_ext_next, 908
 - _M_in_beg, 909
 - _M_in_cur, 909
 - _M_in_end, 909
 - _M_mode, 909
 - _M_out_beg, 910
 - _M_out_cur, 910
 - _M_out_end, 910
 - _M_pback, 910
 - _M_pback_cur_save, 911
 - _M_pback_end_save, 911
 - _M_pback_init, 911
 - _M_reading, 911
 - _M_set_buffer, 892
 - __streambuf_type, 890
- char_type, 890
- close, 892
- eback, 892
- egptr, 893
- epptr, 893
- gbump, 893
- getloc, 894
- gptr, 894
- imbue, 894
- in_avail, 895
- int_type, 891
- is_open, 895
- off_type, 891
- open, 895, 896
- overflow, 896
- pbackfail, 897
- pbase, 897
- pbump, 898
- pos_type, 891
- pptr, 898
- pubimbue, 898
- pubseekoff, 899
- pubseekpos, 899
- pubsetbuf, 899
- pubsync, 899
- sbumpc, 900
- seekoff, 900
- seekpos, 900
- setbuf, 901
- setg, 901
- setp, 902
- sgetc, 902
- sgetn, 902
- showmanyc, 903
- snextc, 903
- sputbackc, 903
- sputc, 904
- sputn, 904
- stossc, 905
- sungetc, 905
- sync, 905
- traits_type, 891
- uflow, 906
- underflow, 906

- xsgetn, 907
- xspu, 907
- __gnu_cxx::encoding_char_traits, 913
- __gnu_cxx::encoding_state, 915
- __gnu_cxx::forced_error, 917
 - what, 917
- __gnu_cxx::free_list, 918
 - _M_clear, 918
 - _M_get, 918
 - _M_insert, 919
- __gnu_cxx::hash_map, 920
- __gnu_cxx::hash_multimap, 923
- __gnu_cxx::hash_multiset, 926
- __gnu_cxx::hash_set, 928
- __gnu_cxx::limit_condition, 930
 - __gnu_cxx::limit_condition::always_ - adjustor, 931
 - __gnu_cxx::limit_condition::limit_ - adjustor, 932
 - __gnu_cxx::limit_condition::never_ - adjustor, 933
- __gnu_cxx::malloc_allocator, 934
- __gnu_cxx::new_allocator, 936
- __gnu_cxx::project1st, 938
 - first_argument_type, 938
 - result_type, 938
 - second_argument_type, 938
- __gnu_cxx::project2nd, 940
 - first_argument_type, 940
 - result_type, 940
 - second_argument_type, 940
- __gnu_cxx::random_condition, 942
 - __gnu_cxx::random_condition::always_ - adjustor, 943
 - __gnu_cxx::random_condition::group_ - adjustor, 944
 - __gnu_cxx::random_condition::never_ - adjustor, 945
- __gnu_cxx::rb_tree, 946
- __gnu_cxx::recursive_init_error, 950
 - what, 950
- __gnu_cxx::rope, 952
- __gnu_cxx::select1st, 959
 - argument_type, 959
 - result_type, 959
- __gnu_cxx::select2nd, 960
 - argument_type, 960
 - result_type, 960
- __gnu_cxx::slist, 961
 - ~stdio_filebuf, 970
 - _M_buf, 992
 - _M_buf_locale, 992
 - _M_buf_size, 992
 - _M_create_pback, 971
 - _M_destroy_pback, 971
 - _M_ext_buf, 992
 - _M_ext_buf_size, 992
 - _M_ext_next, 993
 - _M_in_beg, 993
 - _M_in_cur, 993
 - _M_in_end, 994
 - _M_mode, 994
 - _M_out_beg, 994
 - _M_out_cur, 994
 - _M_out_end, 995
 - _M_pback, 995
 - _M_pback_cur_save, 995
 - _M_pback_end_save, 996
 - _M_pback_init, 996
 - _M_reading, 996
 - _M_set_buffer, 971
 - __streambuf_type, 968
- char_type, 968
- close, 972
- eback, 972
- egptr, 972
- epptr, 973
- fd, 973
- file, 974
- gbump, 974
- getloc, 974
- gptr, 975
- imbue, 975
- in_avail, 976
- int_type, 968
- is_open, 976
- off_type, 968
- open, 976, 977
- overflow, 977
- pbackfail, 978
- pbase, 979

- pbump, 979
- pos_type, 969
- pptr, 980
- pubimbue, 980
- pubseekoff, 981
- pubseekpos, 981
- pubsetbuf, 981
- pubsync, 981
- sbumpc, 982
- seekoff, 982
- seekpos, 983
- setbuf, 983
- setg, 984
- setp, 984
- sgetc, 984
- sgetn, 985
- showmanyc, 985
- snextc, 986
- sputbackc, 986
- sputc, 987
- sputn, 987
- stdio_filebuf, 969, 970
- stoss, 988
- sungetc, 988
- sync, 988
- traits_type, 969
- uflow, 989
- underflow, 989
- xsggetn, 990
- xsgputn, 991
- __gnu_cxx::stdio_sync_filebuf, 998
 - _M_buf_locale, 1019
 - _M_in_beg, 1019
 - _M_in_cur, 1019
 - _M_in_end, 1020
 - _M_out_beg, 1020
 - _M_out_cur, 1020
 - _M_out_end, 1021
 - __streambuf_type, 1001
- char_type, 1001
- eback, 1003
- egptr, 1003
- epptr, 1003
- file, 1004
- gbump, 1004
- getloc, 1004
- gptr, 1005
- imbue, 1005
- in_avail, 1006
- int_type, 1002
- off_type, 1002
- overflow, 1006
- pbackfail, 1007
- pbase, 1007
- pbump, 1008
- pos_type, 1002
- pptr, 1008
- pubimbue, 1009
- pubseekoff, 1009
- pubseekpos, 1009
- pubsetbuf, 1010
- pubsync, 1010
- sbumpc, 1010
- seekoff, 1011
- seekpos, 1011
- setbuf, 1011
- setg, 1012
- setp, 1012
- sgetc, 1013
- sgetn, 1013
- showmanyc, 1013
- snextc, 1014
- sputbackc, 1014
- sputc, 1015
- sputn, 1015
- stoss, 1016
- sungetc, 1016
- sync, 1016
- traits_type, 1002
- uflow, 1017
- underflow, 1017
- xsggetn, 1018
- xsgputn, 1018
- __gnu_cxx::subtractive_rng, 1022
 - argument_type, 1022
 - operator(), 1023
 - result_type, 1022
 - subtractive_rng, 1023
- __gnu_cxx::temporary_buffer, 1024
 - ~temporary_buffer, 1025
 - begin, 1026
 - end, 1026

- requested_size, 1026
- size, 1026
- temporary_buffer, 1025
- __gnu_cxx::throw_allocator_base, 1027
- __gnu_cxx::throw_allocator_limit, 1029
- __gnu_cxx::throw_allocator_random, 1031
- __gnu_cxx::throw_value_base, 1033
- __gnu_cxx::throw_value_limit, 1034
- __gnu_cxx::throw_value_random, 1036
- __gnu_cxx::typelist, 372
 - apply_generator, 372
- __gnu_cxx::unary_compose, 1038
 - argument_type, 1039
 - result_type, 1039
- __gnu_debug, 373
 - __check_dereferenceable, 379
 - __check_singular, 380
 - __check_singular_aux, 380
 - __check_string, 380
 - __valid_range, 380, 381
 - __valid_range_aux, 381
 - __valid_range_aux2, 381, 382
- __gnu_debug::_After_nth_from, 1041
- __gnu_debug::_Not_equal_to, 1042
- __gnu_debug::_Safe_iterator, 1043
 - _M_attach, 1047
 - _M_attach_single, 1047
 - _M_attached_to, 1047
 - _M_dereferenceable, 1048
 - _M_detach, 1048
 - _M_detach_single, 1048
 - _M_get_distance, 1048
 - _M_get_mutex, 1048
 - _M_incrementable, 1049
 - _M_invalidate, 1049
 - _M_invalidate_single, 1049
 - _M_is_begin, 1049
 - _M_is_end, 1050
 - _M_next, 1053
 - _M_prior, 1053
 - _M_sequence, 1053
 - _M_version, 1054
 - _Safe_iterator, 1045, 1046
 - __attribute__, 1046
 - base, 1050
 - operator_iterator, 1050
 - operator*, 1050
 - operator++, 1051
 - operator->, 1052
 - operator--, 1051, 1052
 - operator=, 1052
- __gnu_debug::_Safe_iterator_base, 1055
 - _M_attach, 1057
 - _M_attach_single, 1057
 - _M_attached_to, 1057
 - _M_detach, 1057
 - _M_detach_single, 1057
 - _M_get_mutex, 1057
 - _M_next, 1058
 - _M_prior, 1058
 - _M_sequence, 1058
 - _M_version, 1058
 - _Safe_iterator_base, 1056
 - __attribute__, 1057
- __gnu_debug::_Safe_sequence, 1060
 - _M_const_iterators, 1063
 - _M_detach_all, 1061
 - _M_detach_singular, 1061
 - _M_get_mutex, 1061
 - _M_invalidate_all, 1061
 - _M_invalidate_if, 1062
 - _M_iterators, 1063
 - _M_revalidate_singular, 1062
 - _M_swap, 1062
 - _M_transfer_iter, 1062
 - _M_version, 1063
- __gnu_debug::_Safe_sequence_base, 1064
 - ~_Safe_sequence_base, 1065
 - _M_const_iterators, 1066
 - _M_detach_all, 1065
 - _M_detach_singular, 1065
 - _M_get_mutex, 1065
 - _M_invalidate_all, 1066
 - _M_iterators, 1066
 - _M_revalidate_singular, 1066
 - _M_swap, 1066
 - _M_version, 1067
- __gnu_debug::_is_same, 1040
- __gnu_debug::basic_string, 1068
 - ~basic_string, 1076

- [_M_const_iterators](#), 1123
- [_M_detach_all](#), 1077
- [_M_detach_singular](#), 1077
- [_M_get_mutex](#), 1077
- [_M_invalidate_all](#), 1077
- [_M_invalidate_if](#), 1077
- [_M_iterators](#), 1123
- [_M_revalidate_singular](#), 1078
- [_M_swap](#), 1078
- [_M_transfer_iter](#), 1078
- [_M_version](#), 1124
- [append](#), 1078–1080
- [assign](#), 1081–1084
- [at](#), 1084, 1085
- [basic_string](#), 1075, 1076
- [begin](#), 1085, 1086
- [c_str](#), 1086
- [capacity](#), 1086
- [cbegin](#), 1087
- [cend](#), 1087
- [clear](#), 1087
- [compare](#), 1087–1090
- [copy](#), 1090
- [crbegin](#), 1091
- [crend](#), 1091
- [data](#), 1091
- [empty](#), 1092
- [end](#), 1092
- [erase](#), 1092, 1093
- [find](#), 1094, 1095
- [find_first_not_of](#), 1095–1097
- [find_first_of](#), 1097, 1098
- [find_last_not_of](#), 1099, 1100
- [find_last_of](#), 1100–1102
- [get_allocator](#), 1102
- [insert](#), 1102–1107
- [length](#), 1107
- [max_size](#), 1107
- [npos](#), 1124
- [operator+=](#), 1107–1109
- [operator-=](#), 1109, 1110
- [push_back](#), 1111
- [rbegin](#), 1111, 1112
- [rend](#), 1112
- [replace](#), 1113–1119
- [reserve](#), 1119
- [resize](#), 1120
- [rfind](#), 1120–1122
- [shrink_to_fit](#), 1122
- [size](#), 1122
- [substr](#), 1122
- [swap](#), 1123
- [__gnu_internal](#), 383
- [__gnu_parallel](#), 384
 - [_AlgorithmStrategy](#), 401
 - [_BinIndex](#), 400
 - [_CASable](#), 400
 - [_CASable_bits](#), 447
 - [_CASable_mask](#), 447
 - [_FindAlgorithm](#), 401
 - [_MultiwayMergeAlgorithm](#), 401
 - [_Parallelism](#), 401
 - [_PartialSumAlgorithm](#), 402
 - [_SequenceIndex](#), 400
 - [_SortAlgorithm](#), 402
 - [_SplittingAlgorithm](#), 402
 - [_ThreadIndex](#), 401
 - [__calc_borders](#), 402
 - [__compare_and_swap](#), 402
 - [__compare_and_swap_32](#), 403
 - [__compare_and_swap_64](#), 403
 - [__decode2](#), 404
 - [__determine_samples](#), 404
 - [__encode2](#), 405
 - [__fetch_and_add](#), 405
 - [__fetch_and_add_32](#), 405
 - [__fetch_and_add_64](#), 406
 - [__find_template](#), 406–408
 - [__for_each_template_random_access](#), 409
 - [__for_each_template_random_access_ed](#), 409
 - [__for_each_template_random_access_omp_loop](#), 410
 - [__for_each_template_random_access_omp_loop_static](#), 411
 - [__for_each_template_random_access_workstealing](#), 411
 - [__is_sorted](#), 412
 - [__median_of_three_iterators](#), 413
 - [__merge_advance](#), 413
 - [__merge_advance_movc](#), 414

- [__merge_advance_usual](#), 414
- [__parallel_merge_advance](#), 415, 416
- [__parallel_nth_element](#), 416
- [__parallel_partial_sort](#), 417
- [__parallel_partial_sum](#), 417
- [__parallel_partial_sum_basecase](#), 418
- [__parallel_partial_sum_linear](#), 418
- [__parallel_partition](#), 419
- [__parallel_random_shuffle](#), 420
- [__parallel_random_shuffle_drs](#), 420
- [__parallel_random_shuffle_drs_pu](#), 421
- [__parallel_sort](#), 421–425
- [__parallel_sort_qs](#), 425
- [__parallel_sort_qs_conquer](#), 426
- [__parallel_sort_qs_divide](#), 426
- [__parallel_sort_qsb](#), 427
- [__parallel_unique_copy](#), 427
- [__qsb_conquer](#), 428
- [__qsb_divide](#), 428
- [__qsb_local_sort_with_helping](#), 429
- [__random_number_pow2](#), 429
- [__rd_log2](#), 430
- [__round_up_to_pow2](#), 430
- [__search_template](#), 430
- [__sequential_multiway_merge](#), 431
- [__sequential_random_shuffle](#), 432
- [__shrink](#), 432
- [__shrink_and_double](#), 433
- [__yield](#), 433
- [equally_split](#), 433
- [equally_split_point](#), 434
- [list_partition](#), 434
- [max](#), 435
- [min](#), 435
- [multiseq_partition](#), 435
- [multiseq_selection](#), 436
- [multiway_merge](#), 437
- [multiway_merge_3_variant](#), 439
- [multiway_merge_4_variant](#), 439
- [multiway_merge_exact_splitting](#), 440
- [multiway_merge_loser_tree](#), 441
- [multiway_merge_loser_tree_-sentinel](#), 441
- [multiway_merge_loser_tree_-unguarded](#), 442
- [multiway_merge_sampling_-splitting](#), 443
- [multiway_merge_sentinels](#), 443
- [parallel_multiway_merge](#), 445
- [parallel_sort_mwms](#), 446
- [parallel_sort_mwms_pu](#), 446
- [__gnu_parallel::_DRSSorterPU](#), 1182
- [_M_bins_begin](#), 1182
- [_M_num_threads](#), 1183
- [_M_sd](#), 1183
- [_M_seed](#), 1183
- [__bins_end](#), 1182
- [__gnu_parallel::_-DRandomShufflingGlobalData](#), 1179
- [_DRandomShufflingGlobalData](#), 1180
- [_M_bin_proc](#), 1180
- [_M_dist](#), 1180
- [_M_num_bins](#), 1180
- [_M_num_bits](#), 1180
- [_M_source](#), 1181
- [_M_starts](#), 1181
- [_M_temporaries](#), 1181
- [__gnu_parallel::_DummyReduct](#), 1184
- [__gnu_parallel::_EqualFromLess](#), 1185
- [first_argument_type](#), 1186
- [result_type](#), 1186
- [second_argument_type](#), 1186
- [__gnu_parallel::_EqualTo](#), 1187
- [first_argument_type](#), 1188
- [result_type](#), 1188
- [second_argument_type](#), 1188
- [__gnu_parallel::_GuardedIterator](#), 1189
- [_GuardedIterator](#), 1189
- [operator_RAIter](#), 1190
- [operator<](#), 1191
- [operator<=](#), 1191
- [operator*](#), 1190
- [operator++](#), 1190
- [__gnu_parallel::_IteratorPair](#), 1192
- [first](#), 1193
- [first_type](#), 1193
- [second](#), 1193

- second_type, 1193
- __gnu_parallel::_IteratorTriple, 1195
- __gnu_parallel::_Job, 1197
 - _M_first, 1197
 - _M_last, 1197
 - _M_load, 1197
- __gnu_parallel::_Less, 1199
 - first_argument_type, 1200
 - result_type, 1200
 - second_argument_type, 1200
- __gnu_parallel::_Lexicographic, 1201
 - first_argument_type, 1202
 - result_type, 1202
 - second_argument_type, 1202
- __gnu_parallel::_LexicographicReverse, 1203
 - first_argument_type, 1204
 - result_type, 1204
 - second_argument_type, 1204
- __gnu_parallel::_LoserTree, 1205
 - _M_comp, 1207
 - _M_first_insert, 1207
 - _M_log_k, 1207
 - _M_losers, 1207
 - __delete_min_insert, 1206
 - __get_min_source, 1206
 - __insert_start, 1206
- __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1209
 - _M_comp, 1211
 - _M_first_insert, 1211
 - _M_log_k, 1211
 - _M_losers, 1212
 - __delete_min_insert, 1210
 - __get_min_source, 1210
 - __init_winner, 1210
 - __insert_start, 1210
- __gnu_parallel::_LoserTreeBase, 1213
 - ~_LoserTreeBase, 1214
 - _LoserTreeBase, 1214
 - _M_comp, 1215
 - _M_first_insert, 1215
 - _M_log_k, 1216
 - _M_losers, 1216
 - __get_min_source, 1215
 - __insert_start, 1215
- __gnu_parallel::_LoserTreeBase::_Loser, 1217
 - _M_key, 1217
 - _M_source, 1217
 - _M_sup, 1217
- __gnu_parallel::_LoserTreePointer, 1219
- __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 1221
- __gnu_parallel::_LoserTreePointerBase, 1223
- __gnu_parallel::_-LoserTreePointerBase::_Loser, 1225
- __gnu_parallel::_-LoserTreePointerUnguarded, 1226
- __gnu_parallel::_-LoserTreePointerUnguarded< false, _Tp, _Compare >, 1228
- __gnu_parallel::_-LoserTreePointerUnguardedBase, 1230
- __gnu_parallel::_LoserTreeTraits, 1231
 - _M_use_pointer, 1231
- __gnu_parallel::_LoserTreeUnguarded, 1233
- __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 1235
- __gnu_parallel::_-LoserTreeUnguardedBase, 1237
- __gnu_parallel::_Multiplies, 1239
 - first_argument_type, 1240
 - result_type, 1240
 - second_argument_type, 1240
- __gnu_parallel::_Nothing, 1241
 - operator(), 1241
- __gnu_parallel::_PMWMSortingData, 1245
 - _M_num_threads, 1245
 - _M_offsets, 1245
 - _M_pieces, 1246
 - _M_samples, 1246
 - _M_source, 1246
 - _M_starts, 1246
 - _M_temporary, 1247

- __gnu_parallel::_Piece, 1242
 - _M_begin, 1242
 - _M_end, 1242
- __gnu_parallel::_Plus, 1243
 - first_argument_type, 1244
 - result_type, 1244
 - second_argument_type, 1244
- __gnu_parallel::_PseudoSequence, 1248
 - PseudoSequence, 1248
 - begin, 1249
 - end, 1249
- __gnu_parallel::_-
 - PseudoSequenceIterator, 1250
- __gnu_parallel::_QSBThreadLocal, 1251
 - _M_elements_leftover, 1252
 - _M_global, 1252
 - _M_initial, 1252
 - _M_leftover_parts, 1252
 - _M_num_threads, 1253
 - _Piece, 1251
 - _QSBThreadLocal, 1252
- __gnu_parallel::_RandomNumber, 1254
 - RandomNumber, 1254
 - __genrand_bits, 1255
 - operator(), 1255
- __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1256
 - ~RestrictedBoundedConcurrentQueue, 1257
 - _RestrictedBoundedConcurrentQueue, 1257
 - pop_back, 1257
 - pop_front, 1257
 - push_front, 1257
- __gnu_parallel::_SamplingSorter, 1259
- __gnu_parallel::_SamplingSorter< false,
 - _RAIter, _StrictWeakOrdering
 - >, 1260
- __gnu_parallel::_Settings, 1261
 - __attribute__, 1262
 - accumulate_minimal_n, 1263
 - adjacent_difference_minimal_n, 1263
 - cache_line_size, 1263
 - count_minimal_n, 1263
 - fill_minimal_n, 1263
 - find_increasing_factor, 1263
 - find_initial_block_size, 1263
 - find_maximum_block_size, 1264
 - find_sequential_search_size, 1264
 - for_each_minimal_n, 1264
 - generate_minimal_n, 1264
 - L1_cache_size, 1264
 - L2_cache_size, 1264
 - max_element_minimal_n, 1265
 - merge_minimal_n, 1265
 - merge_oversampling, 1265
 - min_element_minimal_n, 1265
 - multiway_merge_minimal_k, 1265
 - multiway_merge_minimal_n, 1265
 - multiway_merge_oversampling, 1266
 - nth_element_minimal_n, 1266
 - partial_sort_minimal_n, 1266
 - partial_sum_dilation, 1266
 - partial_sum_minimal_n, 1266
 - partition_chunk_share, 1266
 - partition_chunk_size, 1266
 - partition_minimal_n, 1267
 - qsb_steals, 1267
 - random_shuffle_minimal_n, 1267
 - replace_minimal_n, 1267
 - search_minimal_n, 1267
 - set, 1262
 - set_difference_minimal_n, 1267
 - set_intersection_minimal_n, 1267
 - set_symmetric_difference_minimal_n, 1268
 - set_union_minimal_n, 1268
 - sort_minimal_n, 1268
 - sort_mwms_oversampling, 1268
 - sort_qs_num_samples_preset, 1268
 - sort_qsb_base_case_maximal_n, 1268
 - TLB_size, 1269
 - transform_minimal_n, 1269
 - unique_copy_minimal_n, 1269
- __gnu_parallel::_SplitConsistently, 1270
- __gnu_parallel::_SplitConsistently<
 - false, _RAIter, _Compare,

-
- `_SortingPlacesIterator >`, 1271
 - `__gnu_parallel::_SplitConsistently<`
 - `true, _RAIter, _Compare,`
 - `_SortingPlacesIterator >`, 1272
 - `__gnu_parallel::_VoidFunctor`, 1273
 - `__gnu_parallel::__accumulate_binop_-`
 - `reduct`, 1125
 - `__gnu_parallel::__accumulate_selector`, 1126
 - `_M_finish_iterator`, 1127
 - `operator()`, 1126
 - `__gnu_parallel::__adjacent_difference_-`
 - `selector`, 1128
 - `_M_finish_iterator`, 1128
 - `__gnu_parallel::__adjacent_find_-`
 - `selector`, 1130
 - `_M_sequential_algorithm`, 1130
 - `operator()`, 1131
 - `__gnu_parallel::__binder1st`, 1132
 - `argument_type`, 1133
 - `result_type`, 1133
 - `__gnu_parallel::__binder2nd`, 1134
 - `argument_type`, 1135
 - `result_type`, 1135
 - `__gnu_parallel::__count_if_selector`, 1136
 - `_M_finish_iterator`, 1137
 - `operator()`, 1136
 - `__gnu_parallel::__count_selector`, 1138
 - `_M_finish_iterator`, 1139
 - `operator()`, 1138
 - `__gnu_parallel::__fill_selector`, 1140
 - `_M_finish_iterator`, 1141
 - `operator()`, 1140
 - `__gnu_parallel::__find_first_of_selector`, 1142
 - `_M_sequential_algorithm`, 1143
 - `operator()`, 1143
 - `__gnu_parallel::__find_if_selector`, 1144
 - `_M_sequential_algorithm`, 1144
 - `operator()`, 1145
 - `__gnu_parallel::__for_each_selector`, 1146
 - `_M_finish_iterator`, 1147
 - `operator()`, 1146
 - `__gnu_parallel::__generate_selector`, 1148
 - `_M_finish_iterator`, 1149
 - `operator()`, 1148
 - `__gnu_parallel::__generic_find_selector`, 1150
 - `__gnu_parallel::__generic_for_each_-`
 - `selector`, 1151
 - `_M_finish_iterator`, 1152
 - `__gnu_parallel::__identity_selector`, 1153
 - `_M_finish_iterator`, 1154
 - `operator()`, 1153
 - `__gnu_parallel::__inner_product_-`
 - `selector`, 1155
 - `_M_finish_iterator`, 1157
 - `__begin1_iterator`, 1156
 - `__begin2_iterator`, 1157
 - `__inner_product_selector`, 1156
 - `operator()`, 1156
 - `__gnu_parallel::__max_element_reduct`, 1158
 - `__gnu_parallel::__min_element_reduct`, 1159
 - `__gnu_parallel::__mismatch_selector`, 1160
 - `_M_sequential_algorithm`, 1160
 - `operator()`, 1161
 - `__gnu_parallel::__multiway_merge_-`
 - `3_variant_sentinel_switch`, 1162
 - `__gnu_parallel::__multiway_merge_-`
 - `3_variant_sentinel_switch<`
 - `true, _RAIterIterator, _RAIter3,`
 - `_DifferenceTp, _Compare >`, 1163
 - `__gnu_parallel::__multiway_merge_-`
 - `4_variant_sentinel_switch`, 1164
 - `__gnu_parallel::__multiway_merge_-`
 - `4_variant_sentinel_switch<`
 - `true, _RAIterIterator, _RAIter3,`
 - `_DifferenceTp, _Compare >`, 1165
 - `__gnu_parallel::__multiway_merge_-`
 - `k_variant_sentinel_switch`, 1166
-

-
- __gnu_parallel::__multiway_merge_k_variant_sentinel_switch<false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 1167
 - __gnu_parallel::__replace_if_selector, 1168
 - _M_finish_iterator, 1169
 - __new_val, 1169
 - __replace_if_selector, 1169
 - operator(), 1169
 - __gnu_parallel::__replace_selector, 1171
 - _M_finish_iterator, 1172
 - __new_val, 1172
 - __replace_selector, 1171
 - operator(), 1172
 - __gnu_parallel::__transform1_selector, 1173
 - _M_finish_iterator, 1174
 - operator(), 1173
 - __gnu_parallel::__transform2_selector, 1175
 - _M_finish_iterator, 1176
 - operator(), 1175
 - __gnu_parallel::__unary_negate, 1177
 - argument_type, 1178
 - result_type, 1178
 - __gnu_parallel::balanced_quicksort_tag, 1274
 - __get_num_threads, 1274
 - set_num_threads, 1274
 - __gnu_parallel::balanced_tag, 1276
 - __get_num_threads, 1276
 - set_num_threads, 1276
 - __gnu_parallel::constant_size_blocks_tag, 1278
 - __gnu_parallel::default_parallel_tag, 1279
 - __get_num_threads, 1279
 - set_num_threads, 1279
 - __gnu_parallel::equal_split_tag, 1281
 - __gnu_parallel::exact_tag, 1282
 - __get_num_threads, 1282
 - set_num_threads, 1282
 - __gnu_parallel::find_tag, 1284
 - __gnu_parallel::growing_blocks_tag, 1285
 - __gnu_parallel::multiway_mergesort_exact_tag, 1286
 - __get_num_threads, 1286
 - set_num_threads, 1286
 - __gnu_parallel::multiway_mergesort_sampling_tag, 1288
 - __get_num_threads, 1288
 - set_num_threads, 1288
 - __gnu_parallel::multiway_mergesort_tag, 1290
 - __get_num_threads, 1290
 - set_num_threads, 1290
 - __gnu_parallel::omp_loop_static_tag, 1292
 - __get_num_threads, 1292
 - set_num_threads, 1292
 - __gnu_parallel::omp_loop_tag, 1294
 - __get_num_threads, 1294
 - set_num_threads, 1294
 - __gnu_parallel::parallel_tag, 1296
 - __get_num_threads, 1297
 - parallel_tag, 1297
 - set_num_threads, 1297
 - __gnu_parallel::quicksort_tag, 1299
 - __get_num_threads, 1299
 - set_num_threads, 1299
 - __gnu_parallel::sampling_tag, 1301
 - __get_num_threads, 1301
 - set_num_threads, 1301
 - __gnu_parallel::sequential_tag, 1303
 - __gnu_parallel::unbalanced_tag, 1304
 - __get_num_threads, 1304
 - set_num_threads, 1304
 - __gnu_pbds, 448
 - __gnu_pbds::associative_container_tag, 1306
 - __gnu_pbds::basic_hash_table, 1307
 - __gnu_pbds::basic_hash_tag, 1309
 - __gnu_pbds::basic_tree, 1310
 - __gnu_pbds::basic_tree_tag, 1312
 - __gnu_pbds::binary_heap_tag, 1313
 - __gnu_pbds::binomial_heap_tag, 1314
 - __gnu_pbds::cc_hash_table, 1315
 - __gnu_pbds::cc_hash_tag, 1318
-

-
- `__gnu_pbds::container_base`, 1319
 - `__gnu_pbds::container_tag`, 1321
 - `__gnu_pbds::container_traits`, 1322
 - `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`, 1323
 - `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >`, 1324
 - `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >`, 1325
 - `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >`, 1326
 - `__gnu_pbds::gp_hash_table`, 1327
 - `__gnu_pbds::gp_hash_tag`, 1330
 - `__gnu_pbds::list_update`, 1331
 - `__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`, 1337
 - `__gnu_pbds::priority_queue_tag`, 1338
 - `__gnu_pbds::rb_tree_tag`, 1339
 - `__gnu_pbds::rc_binomial_heap_tag`, 1340
 - `__gnu_pbds::sequence_tag`, 1341
 - `__gnu_pbds::splay_tree_tag`, 1342
 - `__gnu_pbds::string_tag`, 1343
 - `__gnu_pbds::thin_heap_tag`, 1344
 - `__gnu_pbds::tree`, 1345
 - `__gnu_pbds::tree_tag`, 1347
 - `__gnu_pbds::trie`, 1348
 - `__gnu_pbds::trie_tag`, 1350
 - `__gnu_profile`, 451
 - `__get__global_lock`, 456
 - `__profcxx_init`, 456
 - `__report`, 456
 - `__gnu_profile::__container_size_info`, 1351
 - `__gnu_profile::__container_size_stack_info`, 1353
 - `__gnu_profile::__hashfunc_info`, 1355
 - `__gnu_profile::__hashfunc_stack_info`, 1357
 - `__gnu_profile::__list2vector_info`, 1359
 - `__gnu_profile::__map2umap_info`, 1361
 - `__gnu_profile::__map2umap_stack_info`, 1363
 - `__gnu_profile::__object_info_base`, 1365
 - `__gnu_profile::__reentrance_guard`, 1366
 - `__gnu_profile::__stack_hash`, 1367
 - `__gnu_profile::__stack_info_base`, 1368
 - `__gnu_profile::__trace_base`, 1369
 - `__gnu_profile::__trace_container_size`, 1370
 - `__gnu_profile::__trace_hash_func`, 1371
 - `__gnu_profile::__trace_hashtable_size`, 1373
 - `__gnu_profile::__trace_map2umap`, 1374
 - `__gnu_profile::__trace_vector_size`, 1376
 - `__gnu_profile::__trace_vector_to_list`, 1377
 - `__gnu_profile::__vector2list_info`, 1379
 - `__gnu_profile::__vector2list_stack_info`, 1381
 - `__gnu_profile::__warning_data`, 1383
 - `__gnu_sequential`, 457
 - `__heap_select`
 - `std`, 602, 603
 - `__init_winner`
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1210
 - `__inner_product_selector`
 - `__gnu_parallel::__inner_product_selector`, 1156
 - `__inplace_stable_partition`
 - `std`, 603
 - `__inplace_stable_sort`
 - `std`, 603
 - `__insert_start`
 - `__gnu_parallel::_LoserTree`, 1206
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1210
 - `__gnu_parallel::_LoserTreeBase`, 1215
 - `__insertion_sort`
 - `std`, 604
 - `__introsort_loop`
 - `std`, 604
 - `__invoke`
-

- std, 605
- __is_sorted
 - __gnu_parallel, 412
- __iterator_category
 - iterators, 283
- __lg
 - std, 605
- __match_flag
 - std::regex_constants, 746
- __median
 - SGIextensions, 16
- __median_of_three_iterators
 - __gnu_parallel, 413
- __merge_adaptive
 - std, 605
- __merge_advance
 - __gnu_parallel, 413
- __merge_advance_movc
 - __gnu_parallel, 414
- __merge_advance_usual
 - __gnu_parallel, 414
- __merge_backward
 - std, 606
- __merge_without_buffer
 - std, 606, 607
- __move_median_first
 - std, 607
- __new_val
 - __gnu_parallel::__replace_if_-
selector, 1169
 - __gnu_parallel::__replace_selector,
1172
- __num_bitmaps
 - __gnu_cxx::__detail, 371
- __num_blocks
 - __gnu_cxx::__detail, 371
- __num_get_type
 - std::basic_fstream, 1683
 - std::basic_ifstream, 1752
 - std::basic_ios, 1806
 - std::basic_iostream, 1841
 - std::basic_istream, 1909
 - std::basic_istream, 1966
 - std::basic_ofstream, 2022
 - std::basic_ostream, 2068
 - std::basic_ostringstream, 2116
- std::basic_stringstream, 2286
- __num_put_type
 - std::basic_fstream, 1683
 - std::basic_ifstream, 1752
 - std::basic_ios, 1806
 - std::basic_iostream, 1841, 1842
 - std::basic_istream, 1909
 - std::basic_istream, 1966
 - std::basic_ofstream, 2022
 - std::basic_ostream, 2068
 - std::basic_ostringstream, 2116
 - std::basic_stringstream, 2286
- __parallel_merge_advance
 - __gnu_parallel, 415, 416
- __parallel_nth_element
 - __gnu_parallel, 416
- __parallel_partial_sort
 - __gnu_parallel, 417
- __parallel_partial_sum
 - __gnu_parallel, 417
- __parallel_partial_sum_basecase
 - __gnu_parallel, 418
- __parallel_partial_sum_linear
 - __gnu_parallel, 418
- __parallel_partition
 - __gnu_parallel, 419
- __parallel_random_shuffle
 - __gnu_parallel, 420
- __parallel_random_shuffle_drs
 - __gnu_parallel, 420
- __parallel_random_shuffle_drs_pu
 - __gnu_parallel, 421
- __parallel_sort
 - __gnu_parallel, 421–425
- __parallel_sort_qs
 - __gnu_parallel, 425
- __parallel_sort_qs_conquer
 - __gnu_parallel, 426
- __parallel_sort_qs_divide
 - __gnu_parallel, 426
- __parallel_sort_qsb
 - __gnu_parallel, 427
- __parallel_unique_copy
 - __gnu_parallel, 427
- __partition
 - std, 607, 608

-
- __profcxx_init
 - __gnu_profile, 456
 - __qsb_conquer
 - __gnu_parallel, 428
 - __qsb_divide
 - __gnu_parallel, 428
 - __qsb_local_sort_with_helping
 - __gnu_parallel, 429
 - __random_number_pow2
 - __gnu_parallel, 429
 - __rd_log2
 - __gnu_parallel, 430
 - __replace_if_selector
 - __gnu_parallel::__replace_if_selector, 1169
 - __replace_selector
 - __gnu_parallel::__replace_selector, 1171
 - __report
 - __gnu_profile, 456
 - __reverse
 - std, 608
 - __rotate
 - std, 608, 609
 - __rotate_adaptive
 - std, 609
 - __round_up_to_pow2
 - __gnu_parallel, 430
 - __search_n
 - std, 609, 610
 - __search_template
 - __gnu_parallel, 430
 - __sequential_multiway_merge
 - __gnu_parallel, 431
 - __sequential_random_shuffle
 - __gnu_parallel, 432
 - __shrink
 - __gnu_parallel, 432
 - __shrink_and_double
 - __gnu_parallel, 433
 - __stable_partition_adaptive
 - std, 610
 - __static_pointer_cast
 - __gnu_cxx, 357
 - __streambuf_type
 - __gnu_cxx::enc_filebuf, 890
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 1001
 - std::basic_filebuf, 1646
 - std::basic_streambuf, 2172
 - std::basic_stringbuf, 2255
 - __syntax_option
 - std::regex_constants, 746
 - __unguarded_insertion_sort
 - std, 611
 - __unguarded_linear_insert
 - std, 611
 - __unguarded_partition
 - std, 612
 - __unguarded_partition_pivot
 - std, 612
 - __unique_copy
 - std, 613, 614
 - __valid_range
 - __gnu_debug, 380, 381
 - __valid_range_aux
 - __gnu_debug, 381
 - __valid_range_aux2
 - __gnu_debug, 381, 382
 - __verbose_terminate_handler
 - exceptions, 35
 - __versa_string
 - __gnu_cxx::__versa_string, 800–803
 - __yield
 - __gnu_parallel, 433
 - a
 - std::extreme_value_distribution, 2603
 - std::weibull_distribution, 3396
 - abi, 458
 - abs
 - complex_numbers, 42
 - accumulate
 - std, 615
 - accumulate_minimal_n
 - __gnu_parallel::_Settings, 1263
 - acos
 - complex_numbers, 43
 - acosh
-

- complex_numbers, 43
- Adaptors for pointers to functions, 269
- Adaptors for pointers to members, 271
- adjacent_difference
 - std, 616
- adjacent_difference_minimal_n
 - __gnu_parallel::_Settings, 1263
- adjacent_find
 - non_mutating_algorithms, 197
- adjustfield
 - std::basic_fstream, 1739
 - std::basic_ifstream, 1795
 - std::basic_ios, 1826
 - std::basic_istream, 1895
 - std::basic_ostream, 1950
 - std::basic_stringstream, 2008
 - std::basic_ofstream, 2056
 - std::basic_ostringstream, 2101
 - std::basic_ostringstream, 2149
 - std::basic_stringstream, 2341
 - std::ios_base, 2747
- advance
 - std, 617
- algo.h, 3399
- algbase.h, 3414
- algorithm, 3416, 3417, 3419
- algorithmfwd.h, 3420, 3427
- Algorithms, 171
- all
 - std::bitset, 2373
 - std::locale, 2855
- all_of
 - non_mutating_algorithms, 197
- allocate_shared
 - pointer_abstractions, 67
 - std::shared_ptr, 3259
- allocator.h, 3440
- allocator_type
 - std::set, 3228
- Allocators, 246
- alpha
 - std::gamma_distribution, 2662
- any
 - std::bitset, 2373
- any_of
 - non_mutating_algorithms, 198
- app
 - std::basic_fstream, 1739
 - std::basic_ifstream, 1795
 - std::basic_ios, 1826
 - std::basic_istream, 1895
 - std::basic_ostream, 1950
 - std::basic_stringstream, 2008
 - std::basic_ofstream, 2056
 - std::basic_ostringstream, 2101
 - std::basic_ostringstream, 2149
 - std::basic_stringstream, 2341
 - std::ios_base, 2748
- append
 - __gnu_cxx::__versa_string, 804–806
 - __gnu_debug::basic_string, 1078–1080
 - std::basic_string, 2202–2205
- apply
 - numeric_arrays, 92
- apply_generator
 - __gnu_cxx::typelist, 372
- arg
 - complex_numbers, 43
- argument_type
 - __gnu_cxx::__detail::_Ffit_finder, 776
 - __gnu_cxx::binary_compose, 875
 - __gnu_cxx::select1st, 959
 - __gnu_cxx::select2nd, 960
 - __gnu_cxx::subtractive_rng, 1022
 - __gnu_cxx::unary_compose, 1039
 - __gnu_parallel::_binder1st, 1133
 - __gnu_parallel::_binder2nd, 1135
 - __gnu_parallel::_unary_negate, 1178
 - std::_Maybe_unary_or_binary_function<_Res, _T1 >, 1555
 - std::binder1st, 2358
 - std::binder2nd, 2360
 - std::const_mem_fun_ref_t, 2456
 - std::const_mem_fun_t, 2458
 - std::hash, 2686
 - std::hash< ::bitset<_Nb > >, 2689
 - std::hash< ::vector< bool, _Alloc > >, 2691

- std::hash< __debug::bitset< _Nb >
 >, 2693
 std::hash< __debug::vector< bool,
 _Alloc > >, 2695
 std::hash< __gnu_cxx::throw_-
 value_limit >, 2697
 std::hash< __gnu_cxx::throw_-
 value_random >, 2699
 std::hash< __profile::bitset< _Nb >
 >, 2701
 std::hash< __profile::vector< bool,
 _Alloc > >, 2703
 std::hash< _Tp * >, 2705
 std::hash< error_code >, 2707
 std::hash< string >, 2709
 std::hash< thread::id >, 2711
 std::hash< u16string >, 2713
 std::hash< u32string >, 2715
 std::hash< wstring >, 2717
 std::logical_not, 2868
 std::mem_fun_ref_t, 2915
 std::mem_fun_t, 2917
 std::negate, 3012
 std::pointer_to_unary_function,
 3147
 std::unary_function, 3335
 std::unary_negate, 3337
 Arithmetic Classes, 264
 array, 3441, 3442
 array_allocator.h, 3444
 asin
 complex_numbers, 43
 asinh
 complex_numbers, 43
 assign
 __gnu_cxx::__versa_string, 807–
 810
 __gnu_debug::basic_string, 1081–
 1084
 std::basic_regex, 2162–2164
 std::basic_string, 2205–2208
 std::deque, 2555, 2556
 std::forward_list, 2618, 2619
 std::list, 2832
 std::vector, 3373, 3374
 assoc_container.hpp, 3445
 assoc_laguerre
 tr1_math_spec_func, 119
 assoc_legendre
 tr1_math_spec_func, 119
 Associative Containers, 28
 at
 __gnu_cxx::__versa_string, 810,
 811
 __gnu_debug::basic_string, 1084,
 1085
 std::basic_string, 2209
 std::deque, 2556, 2557
 std::map, 2883
 std::vector, 3375
 atan
 complex_numbers, 44
 atanh
 complex_numbers, 44
 ate
 std::basic_fstream, 1739
 std::basic_ifstream, 1795
 std::basic_ios, 1826
 std::basic_iostream, 1896
 std::basic_istream, 1950
 std::basic_istreamstream, 2009
 std::basic_ofstream, 2056
 std::basic_ostream, 2101
 std::basic_ostreamstream, 2149
 std::basic_stringstream, 2341
 std::ios_base, 2748
 atomic, 3447
 atomic_0.h, 3452
 atomic_2.h, 3453
 atomic_base.h, 3454
 atomic_char
 atomics, 254
 atomic_char16_t
 atomics, 254
 atomic_char32_t
 atomics, 254
 atomic_int
 atomics, 254
 atomic_llong
 atomics, 254
 atomic_long
 atomics, 254

- atomic_schar
 - atomics, 255
- atomic_short
 - atomics, 255
- atomic_uchar
 - atomics, 255
- atomic_uint
 - atomics, 255
- atomic_ullong
 - atomics, 255
- atomic_ulong
 - atomics, 255
- atomic_ushort
 - atomics, 255
- atomic_wchar_t
 - atomics, 256
- atomic_word.h, 3456
- atomicfwd_c.h, 3457
- atomicfwd_cxx.h, 3459
- atomicity.h, 3460
- Atomics, 248
- atomics
 - _GLIBCXX_ATOMIC_-
PROPERTY, 254
 - atomic_char, 254
 - atomic_char16_t, 254
 - atomic_char32_t, 254
 - atomic_int, 254
 - atomic_llong, 254
 - atomic_long, 254
 - atomic_schar, 255
 - atomic_short, 255
 - atomic_uchar, 255
 - atomic_uint, 255
 - atomic_ullong, 255
 - atomic_ulong, 255
 - atomic_ushort, 255
 - atomic_wchar_t, 256
 - kill_dependency, 256
 - memory_order, 256
- auto_ptr
 - std::auto_ptr, 1627, 1628
- auto_ptr.h, 3461
- awk
 - std::regex_constants, 748
- b
 - std::extreme_value_distribution, 2603
 - std::weibull_distribution, 3396
- back
 - __gnu_cxx::__versa_string, 811
 - std::deque, 2557
 - std::list, 2833
 - std::queue, 3163
 - std::vector, 3375, 3376
- back_insert_iterator
 - std::back_insert_iterator, 1635
- back_inserter
 - iterators, 283
- bad
 - std::basic_fstream, 1690
 - std::basic_ifstream, 1758
 - std::basic_ios, 1812
 - std::basic_iostream, 1847
 - std::basic_istream, 1914
 - std::basic_istream, 1972
 - std::basic_ofstream, 2028
 - std::basic_ostream, 2074
 - std::basic_ostringstream, 2122
 - std::basic_stringstream, 2292
- badbit
 - std::basic_fstream, 1739
 - std::basic_ifstream, 1795
 - std::basic_ios, 1827
 - std::basic_iostream, 1896
 - std::basic_istream, 1950
 - std::basic_istream, 2009
 - std::basic_ofstream, 2056
 - std::basic_ostream, 2101
 - std::basic_ostringstream, 2150
 - std::basic_stringstream, 2341
 - std::ios_base, 2748
- balanced_quicksort.h, 3462
- base
 - __gnu_debug::_Safe_iterator, 1050
 - std::discard_block_engine, 2574
 - std::independent_bits_engine, 2722
 - std::reverse_iterator, 3217
 - std::shuffle_order_engine, 3263
- base.h, 3464, 3466
- basefield

- std::basic_fstream, 1739
- std::basic_ifstream, 1796
- std::basic_ios, 1827
- std::basic_iostream, 1896
- std::basic_istream, 1951
- std::basic_istreamstream, 2009
- std::basic_ofstream, 2056
- std::basic_ostream, 2102
- std::basic_ostringstream, 2150
- std::basic_stringstream, 2341
- std::ios_base, 2748
- basic
 - std::regex_constants, 748
- basic_file.h, 3467
- basic_filebuf
 - std::basic_filebuf, 1648
- basic_fstream
 - std::basic_fstream, 1688
- basic_ifstream
 - std::basic_ifstream, 1757
- basic_ios
 - std::basic_ios, 1811
- basic_ios.h, 3468
- basic_ios.tcc, 3469
- basic_iostream
 - std::basic_iostream, 1846
- basic_istream
 - std::basic_istream, 1914
- basic_istreamstream
 - std::basic_istreamstream, 1971
- basic_iterator.h, 3470
- basic_ofstream
 - std::basic_ofstream, 2026, 2027
- basic_ostream
 - std::basic_ostream, 2073
- basic_ostringstream
 - std::basic_ostringstream, 2120
- basic_regex
 - std::basic_regex, 2159–2161
- basic_streambuf
 - std::basic_streambuf, 2173
- basic_string
 - __gnu_debug::basic_string, 1075, 1076
 - std::basic_string, 2198–2201
- basic_string.h, 3471
- basic_string.tcc, 3475
- basic_stringbuf
 - std::basic_stringbuf, 2257
- basic_stringstream
 - std::basic_stringstream, 2291
- basic_types.hpp, 3477
- before_begin
 - std::forward_list, 2619
- beg
 - std::basic_fstream, 1740
 - std::basic_ifstream, 1796
 - std::basic_ios, 1827
 - std::basic_iostream, 1896
 - std::basic_istream, 1951
 - std::basic_istreamstream, 2009
 - std::basic_ofstream, 2057
 - std::basic_ostream, 2102
 - std::basic_ostringstream, 2150
 - std::basic_stringstream, 2342
 - std::ios_base, 2748
- begin
 - __gnu_cxx::__versa_string, 812
 - __gnu_cxx::temporary_buffer, 1026
 - __gnu_debug::basic_string, 1085, 1086
 - __gnu_parallel::_PseudoSequence, 1249
 - std::_Temporary_buffer, 1578
 - std::basic_string, 2210
 - std::deque, 2558
 - std::forward_list, 2620
 - std::list, 2833
 - std::map, 2883
 - std::match_results, 2905
 - std::multimap, 2975
 - std::multiset, 2996
 - std::set, 3234
 - std::vector, 3376
- Bernoulli Distributions, 297
- bernoulli_distribution
 - std::bernoulli_distribution, 2349
- beta
 - std::gamma_distribution, 2662
- tr1_math_spec_func, 119
- binary
 - std::basic_fstream, 1740

- std::basic_ifstream, 1796
- std::basic_ios, 1827
- std::basic_iostream, 1897
- std::basic_istream, 1951
- std::basic_istream, 2010
- std::basic_ofstream, 2057
- std::basic_ostream, 2102
- std::basic_ostringstream, 2150
- std::basic_stringstream, 2342
- std::ios_base, 2749
- Binary Search Algorithms, 240
- binary_search
 - binary_search_algorithms, 241
- binary_search_algorithms
 - binary_search, 241
 - equal_range, 242
 - lower_bound, 243, 244
 - upper_bound, 244, 245
- bind
 - binders, 169
 - std, 617
- bind1st
 - binders, 169
- bind2nd
 - binders, 170
- Binder Classes, 168
- binders
 - bind, 169
 - bind1st, 169
 - bind2nd, 170
- binders.h, 3478
- bitmap_allocator.h, 3479
 - _BALLOC_ALIGN_BYTES, 3480
- bitset, 3481, 3483, 3484
 - std::bitset, 2371, 2372
- boolalpha
 - std, 618
 - std::basic_fstream, 1740
 - std::basic_ifstream, 1796
 - std::basic_ios, 1828
 - std::basic_iostream, 1897
 - std::basic_istream, 1951
 - std::basic_istream, 2010
 - std::basic_ofstream, 2057
 - std::basic_ostream, 2102
 - std::basic_ostringstream, 2151
 - std::basic_stringstream, 2342
 - std::ios_base, 2749
- Boolean Operations Classes, 266
- boost_concept_check.h, 3485
- boost_sp_counted_base.h, 3486
- c
 - std::queue, 3164
- c++0x_warning.h, 3487
- c++allocator.h, 3488
- c++config.h, 3489
- c++io.h, 3494
- c++locale.h, 3495
- c++locale_internal.h, 3496
- c_str
 - __gnu_cxx::__versa_string, 812
 - __gnu_debug::basic_string, 1086
 - std::basic_string, 2210
- cache_line_size
 - __gnu_parallel::_Settings, 1263
- call_once
 - mutexes, 70
- capacity
 - __gnu_cxx::__versa_string, 812
 - __gnu_debug::basic_string, 1086
 - std::basic_string, 2210
 - std::vector, 3376
- cassert, 3497
- category
 - std::locale, 2851
- cbefore_begin
 - std::forward_list, 2620
- cbegin
 - __gnu_cxx::__versa_string, 813
 - __gnu_debug::basic_string, 1087
 - std::basic_string, 2211
 - std::deque, 2558
 - std::forward_list, 2620
 - std::list, 2833
 - std::map, 2884
 - std::match_results, 2905
 - std::multimap, 2975
 - std::multiset, 2996
 - std::set, 3234
 - std::vector, 3377
- ccomplex, 3498, 3499

- cctype, 3500–3502
cend
 __gnu_cxx::__versa_string, 813
 __gnu_debug::basic_string, 1087
 std::basic_string, 2211
 std::deque, 2558
 std::forward_list, 2621
 std::list, 2834
 std::map, 2884
 std::match_results, 2905
 std::multimap, 2975
 std::multiset, 2996
 std::set, 3234
 std::vector, 3377
cerr
 std, 685
cerrno, 3503
cfenv, 3504–3506
cfloat, 3507, 3508
char_traits.h, 3509
char_type
 __gnu_cxx::enc_filebuf, 890
 __gnu_cxx::stdio_filebuf, 968
 __gnu_cxx::stdio_sync_filebuf,
 1001
 std::__ctype_abstract_base, 1394
 std::basic_filebuf, 1647
 std::basic_fstream, 1684
 std::basic_ifstream, 1753
 std::basic_ios, 1806
 std::basic_iostream, 1842
 std::basic_istream, 1909
 std::basic_istreamstream, 1967
 std::basic_ofstream, 2022
 std::basic_ostream, 2069
 std::basic_ostreamstream, 2116
 std::basic_streambuf, 2172
 std::basic_stringbuf, 2255
 std::basic_stringstream, 2287
 std::collate, 2434
 std::collate_byname, 2441
 std::ctype, 2462
 std::ctype< char >, 2476
 std::ctype< wchar_t >, 2491
 std::ctype_byname, 2506
 std::ctype_byname< char >, 2521
 std::istreambuf_iterator, 2803
 std::messages, 2920
 std::messages_byname, 2925
 std::money_get, 2933
 std::money_put, 2939
 std::moneypunct, 2946
 std::moneypunct_byname, 2959
 std::num_get, 3029
 std::num_put, 3041
 std::numpunct, 3095
 std::numpunct_byname, 3103
 std::ostream_iterator, 3110
 std::ostreambuf_iterator, 3115
 std::time_get, 3292
 std::time_get_byname, 3303
 std::time_put, 3313
 std::time_put_byname, 3318
checkers.h, 3510
chrono, 3511
cin
 std, 685
cinttypes, 3515–3517
ciso646, 3518
classic
 std::locale, 2853
classic_table
 std::ctype< char >, 2477
 std::ctype_byname< char >, 2522
clear
 __gnu_cxx::__versa_string, 813
 __gnu_debug::basic_string, 1087
 std::basic_fstream, 1690
 std::basic_ifstream, 1758
 std::basic_ios, 1812
 std::basic_iostream, 1847
 std::basic_istream, 1915
 std::basic_istreamstream, 1972
 std::basic_ofstream, 2028
 std::basic_ostream, 2074
 std::basic_ostreamstream, 2122
 std::basic_string, 2211
 std::basic_stringstream, 2293
 std::deque, 2558
 std::forward_list, 2621
 std::list, 2834
 std::map, 2884

- std::multimap, 2976
- std::multiset, 2996
- std::set, 3234
- std::vector, 3377
- climits, 3519, 3520
- locale, 3521
- clog
 - std, 685
- close
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 972
 - std::basic_filebuf, 1649
 - std::basic_fstream, 1690
 - std::basic_ifstream, 1759
 - std::basic_ofstream, 2029
- cmath, 3522, 3526, 3530
- cmath.tcc, 3531
- code
 - std::regex_error, 3189
- codecvt.h, 3532
- codecvt_specializations.h, 3534
- collate
 - std::collate, 2434
 - std::locale, 2855
 - std::regex_constants, 748
- combine
 - std::locale, 2853
- comp_ellint_1
 - tr1_math_spec_func, 119
- comp_ellint_2
 - tr1_math_spec_func, 119
- comp_ellint_3
 - tr1_math_spec_func, 119
- compare
 - __gnu_cxx::__versa_string, 813–817
 - __gnu_debug::basic_string, 1087–1090
 - std::basic_string, 2211–2215
 - std::collate, 2435
 - std::collate_byname, 2441
 - std::sub_match, 3281
- Comparison Classes, 265
- compatibility.h, 3535, 3536
- compiletime_settings.h, 3537
 - _GLIBCXX_ASSERTIONS, 3537
 - _GLIBCXX_CALL, 3537
 - _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_L1, 3538
 - _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_TLB, 3538
 - _GLIBCXX_SCALE_DOWN_-FPU, 3538
 - _GLIBCXX_VERBOSE_LEVEL, 3538
- complex, 3539, 3543, 3544
 - std::complex, 2446
- Complex Numbers, 38
- complex.h, 3546
- complex_numbers
 - abs, 42
 - acos, 43
 - acosh, 43
 - arg, 43
 - asin, 43
 - asinh, 43
 - atan, 44
 - atanh, 44
 - conj, 44
 - cos, 44
 - cosh, 44
 - exp, 44
 - fabs, 45
 - log, 45
 - log10, 45
 - norm, 45
 - operator<<, 49
 - operator>>, 50
 - operator*, 46
 - operator*=: 46, 47
 - operator+, 47
 - operator+=, 47
 - operator-, 48
 - operator-=, 48
 - operator/, 48, 49
 - operator/=, 49
 - operator=, 50
 - operator==, 50
 - polar, 51
 - pow, 51

- sin, 51
- sinh, 52
- sqrt, 52
- tan, 52
- tanh, 52
- compose1
 - SGIextensions, 18
- compose2
 - SGIextensions, 18
- concept_check.h, 3547
- concurrency.h, 3548
- Concurrency, 31
- cond_dealtor.hpp, 3549
- Condition Variables, 53
- condition_variable, 3550
- condition_variables
 - cv_status, 53
- conf_hyperg
 - tr1_math_spec_func, 120
- conj
 - complex_numbers, 44
- const_iterator
 - std::set, 3228
- const_pointer
 - std::set, 3228
- const_reference
 - std::set, 3229
- const_reverse_iterator
 - std::set, 3229
- constant0
 - SGIextensions, 18
- constant1
 - SGIextensions, 18
- constant2
 - SGIextensions, 18
- constructors_destructor_fn_imps.hpp,
3551
- container_base_dispatch.hpp, 3552
- container_type
 - std::back_insert_iterator, 1634
 - std::front_insert_iterator, 2638
 - std::insert_iterator, 2730
- Containers, 25
- copy
 - __gnu_cxx::__versa_string, 817
 - __gnu_debug::basic_string, 1090
 - mutating_algorithms, 174
 - std::basic_string, 2215
- copy_backward
 - mutating_algorithms, 175
- copy_exception
 - exceptions, 35
- copy_if
 - mutating_algorithms, 175
- copy_n
 - mutating_algorithms, 176
 - SGIextensions, 19
- copyfmt
 - std::basic_fstream, 1690
 - std::basic_ifstream, 1759
 - std::basic_ios, 1812
 - std::basic_iostream, 1848
 - std::basic_istream, 1915
 - std::basic_istreamstream, 1973
 - std::basic_ofstream, 2029
 - std::basic_ostream, 2075
 - std::basic_ostreamstream, 2123
 - std::basic_stringstream, 2293
- cos
 - complex_numbers, 44
- cosh
 - complex_numbers, 44
- count
 - non_mutating_algorithms, 198
 - std::bitset, 2374
 - std::map, 2884
 - std::multimap, 2976
 - std::multiset, 2996
 - std::set, 3234
- count_if
 - non_mutating_algorithms, 199
- count_minimal_n
 - __gnu_parallel::_Settings, 1263
- cout
 - std, 686
- cpp_type_traits.h, 3553
- cpu_defines.h, 3554
- crbegin
 - __gnu_cxx::__versa_string, 818
 - __gnu_debug::basic_string, 1091
 - std::basic_string, 2216
 - std::deque, 2559

- std::list, 2834
- std::map, 2885
- std::multimap, 2976
- std::multiset, 2997
- std::set, 3235
- std::vector, 3377
- ceref
 - std, 618
- cregex_token_iterator
 - tr1_regex, 130
- crend
 - __gnu_cxx::__versa_string, 818
 - __gnu_debug::basic_string, 1091
 - std::basic_string, 2216
 - std::deque, 2559
 - std::list, 2834
 - std::map, 2885
 - std::multimap, 2976
 - std::multiset, 2997
 - std::set, 3235
 - std::vector, 3378
- csetjmp, 3555
- cshift
 - numeric_arrays, 92
- csignal, 3556
- cstdarg, 3557, 3558
- cstdbool, 3559, 3560
- cstddef, 3561
- cstdlib, 3562–3564
- cstdio, 3565–3567
- cstdlib, 3568–3570
- cstring, 3571
- csub_match
 - tr1_regex, 130
- ctgmath, 3572, 3573
- ctime, 3574, 3575
- ctype
 - std::ctype< char >, 2477
 - std::ctype< wchar_t >, 2491
 - std::locale, 2855
- ctype_base.h, 3576
- ctype_inline.h, 3577
- ctype_noninline.h, 3578
- cur
 - std::basic_fstream, 1740
 - std::basic_ifstream, 1796
 - std::basic_ios, 1828
 - std::basic_istream, 1897
 - std::basic_ostream, 2102
 - std::basic_ostringstream, 2151
 - std::basic_stringstream, 2342
 - std::ios_base, 2749
- curr_symbol
 - std::moneypunct, 2948
 - std::moneypunct_byname, 2960
- current_exception
 - exceptions, 35
- cv_status
 - condition_variables, 53
- cwchar, 3579–3581
- cwctype, 3582–3584
- cxxabi-forced.h, 3585
- cxxabi.h, 3586
- cxxabi_tweaks.h, 3588
- cyl_bessel_i
 - tr1_math_spec_func, 120
- cyl_bessel_j
 - tr1_math_spec_func, 120
- cyl_bessel_k
 - tr1_math_spec_func, 120
- cyl_neumann
 - tr1_math_spec_func, 120
- data
 - __gnu_cxx::__versa_string, 818
 - __gnu_debug::basic_string, 1091
 - std::basic_string, 2216
 - std::vector, 3378
- date_order
 - std::time_get, 3293
 - std::time_get_byname, 3303
- debug.h, 3589
- debug_allocator.h, 3590
- debug_map_base.hpp, 3591
- dec
 - std, 618
 - std::basic_fstream, 1740
 - std::basic_ifstream, 1797
 - std::basic_ios, 1828

- std::basic_istream, 1897
- std::basic_istream, 1952
- std::basic_istream, 2010
- std::basic_ofstream, 2057
- std::basic_ostream, 2103
- std::basic_ostringstream, 2151
- std::basic_stringstream, 2342
- std::ios_base, 2749
- decimal, 3592
- Decimal Floating-Point Arithmetic, 167
- decimal128
 - std::decimal::decimal128, 2534
- decimal32
 - std::decimal::decimal32, 2536
- decimal32_to_long_long
 - std::decimal, 742
- decimal64
 - std::decimal::decimal64, 2538
- decimal_point
 - std::moneypunct, 2948
 - std::moneypunct_byname, 2960
 - std::numpunct, 3096
 - std::numpunct_byname, 3103
- denorm_absent
 - std, 600
- denorm_indeterminate
 - std, 600
- denorm_present
 - std, 600
- denorm_min
 - std::numeric_limits, 3050
- densities
 - std::piecewise_constant_distribution, 3131
 - std::piecewise_linear_distribution, 3137
- deque, 3604, 3605, 3607
 - std::deque, 2548–2550
- deque.tcc, 3609
- Diagnostics, 30
- difference_type
 - std::back_insert_iterator, 1634
 - std::front_insert_iterator, 2638
 - std::insert_iterator, 2730
 - std::istream_iterator, 2800
 - std::istreambuf_iterator, 2803
 - std::iterator, 2807
 - std::ostream_iterator, 3110
 - std::ostreambuf_iterator, 3115
 - std::raw_storage_iterator, 3182
 - std::reverse_iterator, 3215
 - std::set, 3229
- digits
 - std::__numeric_limits_base, 1486
 - std::numeric_limits, 3052
- digits10
 - std::__numeric_limits_base, 1486
 - std::numeric_limits, 3052
- discard
 - std::discard_block_engine, 2574
 - std::independent_bits_engine, 2722
 - std::linear_congruential_engine, 2820
 - std::shuffle_order_engine, 3263
- discard_block_engine
 - std::discard_block_engine, 2572, 2573
- distance
 - SGIextensions, 19
 - std, 618
- do_compare
 - std::collate, 2435
 - std::collate_byname, 2441
- do_curr_symbol
 - std::moneypunct, 2948
 - std::moneypunct_byname, 2960
- do_date_order
 - std::time_get, 3293
 - std::time_get_byname, 3304
- do_decimal_point
 - std::moneypunct, 2949
 - std::moneypunct_byname, 2961
 - std::numpunct, 3097
 - std::numpunct_byname, 3103
- do_falsename
 - std::numpunct, 3097
 - std::numpunct_byname, 3104
- do_frac_digits
 - std::moneypunct, 2949
 - std::moneypunct_byname, 2961
- do_get
 - std::money_get, 2935

- std::num_get, 3030–3032
- do_get_date
 - std::time_get, 3294
 - std::time_get_byname, 3304
- do_get_monthname
 - std::time_get, 3294
 - std::time_get_byname, 3305
- do_get_time
 - std::time_get, 3295
 - std::time_get_byname, 3305
- do_get_weekday
 - std::time_get, 3295
 - std::time_get_byname, 3306
- do_get_year
 - std::time_get, 3296
 - std::time_get_byname, 3306
- do_grouping
 - std::moneypunct, 2950
 - std::moneypunct_byname, 2961
 - std::numpunct, 3097
 - std::numpunct_byname, 3104
- do_hash
 - std::collate, 2436
 - std::collate_byname, 2442
- do_is
 - std::__ctype_abstract_base, 1395
 - std::ctype, 2462
 - std::ctype< wchar_t >, 2492
 - std::ctype_byname, 2507
- do_narrow
 - std::__ctype_abstract_base, 1395, 1396
 - std::ctype, 2463
 - std::ctype< char >, 2477, 2478
 - std::ctype< wchar_t >, 2492, 2493
 - std::ctype_byname, 2507, 2508
 - std::ctype_byname< char >, 2522
- do_neg_format
 - std::moneypunct, 2950
 - std::moneypunct_byname, 2962
- do_negative_sign
 - std::moneypunct, 2950
 - std::moneypunct_byname, 2962
- do_out
 - std::__codecvt_abstract_base, 1388
 - std::codecvt, 2403
- std::codecvt< _InternT, _ExternT, encoding_state >, 2409
- std::codecvt< char, char, mbstate_t >, 2415
- std::codecvt< wchar_t, char, mbstate_t >, 2421
- std::codecvt_byname, 2428
- do_pos_format
 - std::moneypunct, 2951
 - std::moneypunct_byname, 2962
- do_positive_sign
 - std::moneypunct, 2951
 - std::moneypunct_byname, 2963
- do_put
 - std::money_put, 2941
 - std::num_put, 3042, 3043
 - std::time_put, 3314
 - std::time_put_byname, 3319
- do_scan_is
 - std::__ctype_abstract_base, 1396
 - std::ctype, 2464
 - std::ctype< wchar_t >, 2493
 - std::ctype_byname, 2508
- do_scan_not
 - std::__ctype_abstract_base, 1397
 - std::ctype, 2464
 - std::ctype< wchar_t >, 2494
 - std::ctype_byname, 2509
- do_thousands_sep
 - std::moneypunct, 2952
 - std::moneypunct_byname, 2963
 - std::numpunct, 3098
 - std::numpunct_byname, 3104
- do_tolower
 - std::__ctype_abstract_base, 1397, 1398
 - std::ctype, 2465
 - std::ctype< char >, 2478, 2479
 - std::ctype< wchar_t >, 2494, 2495
 - std::ctype_byname, 2509, 2510
 - std::ctype_byname< char >, 2523
- do_toupper
 - std::__ctype_abstract_base, 1398, 1399
 - std::ctype, 2465, 2466
 - std::ctype< char >, 2479, 2480

- std::ctype< wchar_t >, 2495, 2496
- std::ctype_byname, 2510, 2511
- std::ctype_byname< char >, 2523, 2524
- do_transform
 - std::collate, 2436
 - std::collate_byname, 2442
- do_truename
 - std::numpunct, 3098
 - std::numpunct_byname, 3105
- do_widen
 - std::__ctype_abstract_base, 1399, 1400
 - std::ctype, 2466, 2467
 - std::ctype< char >, 2480
 - std::ctype< wchar_t >, 2496, 2497
 - std::ctype_byname, 2511, 2512
 - std::ctype_byname< char >, 2524, 2525
- duration_cast
 - std::chrono, 731
- eback
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 972
 - __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1650
 - std::basic_streambuf, 2174
 - std::basic_stringbuf, 2257
- ECMAScript
 - std::regex_constants, 749
- egptr
 - __gnu_cxx::enc_filebuf, 893
 - __gnu_cxx::stdio_filebuf, 972
 - __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1650
 - std::basic_streambuf, 2174
 - std::basic_stringbuf, 2258
- egrep
 - std::regex_constants, 749
- element_type
 - std::auto_ptr, 1627
- ellint_1
 - tr1_math_spec_func, 121
- ellint_2
 - tr1_math_spec_func, 121
- ellint_3
 - tr1_math_spec_func, 121
- emplace
 - std::deque, 2559
 - std::list, 2834
 - std::vector, 3378
- emplace_after
 - std::forward_list, 2621
- emplace_front
 - std::forward_list, 2622
- empty
 - __gnu_cxx::__versa_string, 819
 - __gnu_debug::basic_string, 1092
 - std::basic_string, 2217
 - std::deque, 2560
 - std::forward_list, 2622
 - std::list, 2835
 - std::map, 2885
 - std::match_results, 2905
 - std::multimap, 2977
 - std::multiset, 2997
 - std::priority_queue, 3155
 - std::queue, 3163
 - std::set, 3235
 - std::stack, 3271
 - std::vector, 3379
- enc_filebuf.h, 3610
- end
 - __gnu_cxx::__versa_string, 819
 - __gnu_cxx::temporary_buffer, 1026
 - __gnu_debug::basic_string, 1092
 - __gnu_parallel::_PseudoSequence, 1249
 - std::_Temporary_buffer, 1578
 - std::basic_fstream, 1741
 - std::basic_ifstream, 1797
 - std::basic_ios, 1828
 - std::basic_iostream, 1897
 - std::basic_istream, 1952
 - std::basic_istream, 2010
 - std::basic_ofstream, 2058
 - std::basic_ostream, 2103
 - std::basic_ostringstream, 2151
 - std::basic_string, 2217

- std::basic_stringstream, 2343
- std::deque, 2560
- std::forward_list, 2622
- std::ios_base, 2749
- std::list, 2835
- std::map, 2886
- std::match_results, 2906
- std::multimap, 2977
- std::multiset, 2997
- std::set, 3235
- std::vector, 3379
- endl
 - std, 619
- ends
 - std, 619
- eof
 - std::basic_fstream, 1691
 - std::basic_ifstream, 1759
 - std::basic_ios, 1813
 - std::basic_iostream, 1848
 - std::basic_istream, 1915
 - std::basic_istream, 1973
 - std::basic_ofstream, 2029
 - std::basic_ostream, 2075
 - std::basic_ostringstream, 2123
 - std::basic_stringstream, 2293
- eofbit
 - std::basic_fstream, 1741
 - std::basic_ifstream, 1797
 - std::basic_ios, 1828
 - std::basic_iostream, 1898
 - std::basic_istream, 1952
 - std::basic_istream, 2011
 - std::basic_ofstream, 2058
 - std::basic_ostream, 2103
 - std::basic_ostringstream, 2151
 - std::basic_stringstream, 2343
 - std::ios_base, 2750
- epptr
 - __gnu_cxx::enc_filebuf, 893
 - __gnu_cxx::stdio_filebuf, 973
 - __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1650
 - std::basic_streambuf, 2175
 - std::basic_stringbuf, 2258
- epsilon
 - std::numeric_limits, 3050
- equal
 - non_mutating_algorithms, 199, 200
 - std::istreambuf_iterator, 2805
- equal_range
 - binary_search_algorithms, 242
 - std::map, 2886, 2887
 - std::multimap, 2977, 2978
 - std::multiset, 2997, 2998
 - std::set, 3235, 3236
- equally_split
 - __gnu_parallel, 433
- equally_split.h, 3611
- equally_split_point
 - __gnu_parallel, 434
- erase
 - __gnu_cxx::__versa_string, 819, 820
 - __gnu_debug::basic_string, 1092, 1093
 - std::basic_string, 2217, 2218
 - std::deque, 2560, 2561
 - std::list, 2836
 - std::map, 2887, 2888
 - std::multimap, 2978, 2979
 - std::multiset, 2998, 2999
 - std::set, 3236, 3237
 - std::vector, 3379, 3380
- erase_after
 - std::forward_list, 2623
- error_backref
 - std::regex_constants, 746
- error_badbrace
 - std::regex_constants, 746
- error_badrepeat
 - std::regex_constants, 747
- error_brace
 - std::regex_constants, 747
- error_brack
 - std::regex_constants, 747
- error_collate
 - std::regex_constants, 747
- error_complexity
 - std::regex_constants, 747
- error_constants.h, 3612

- error_ctype
 - std::regex_constants, 747
- error_escape
 - std::regex_constants, 747
- error_paren
 - std::regex_constants, 747
- error_range
 - std::regex_constants, 748
- error_space
 - std::regex_constants, 748
- error_stack
 - std::regex_constants, 748
- error_type
 - std::regex_constants, 746
- event
 - std::basic_fstream, 1687
 - std::basic_ifstream, 1756
 - std::basic_ios, 1811
 - std::basic_iostream, 1846
 - std::basic_istream, 1913
 - std::basic_istream, 1970
 - std::basic_ofstream, 2026
 - std::basic_ostream, 2072
 - std::basic_ostringstream, 2120
 - std::basic_stringstream, 2290
 - std::ios_base, 2741
- event_callback
 - std::basic_fstream, 1684
 - std::basic_ifstream, 1753
 - std::basic_ios, 1807
 - std::basic_iostream, 1842
 - std::basic_istream, 1910
 - std::basic_istream, 1967
 - std::basic_ofstream, 2022
 - std::basic_ostream, 2069
 - std::basic_ostringstream, 2116
 - std::basic_stringstream, 2287
 - std::ios_base, 2739
- exception, 3614
- exception.hpp, 3615
- exception_ptr.h, 3616
- Exceptions, 32
- exceptions
 - __verbose_terminate_handler, 35
 - copy_exception, 35
 - current_exception, 35
 - rethrow_exception, 35
 - rethrow_if_nested, 35
 - set_terminate, 36
 - set_unexpected, 36
 - std::basic_fstream, 1691, 1692
 - std::basic_ifstream, 1760
 - std::basic_ios, 1813, 1814
 - std::basic_iostream, 1849
 - std::basic_istream, 1916
 - std::basic_istream, 1974
 - std::basic_ofstream, 2030
 - std::basic_ostream, 2075, 2076
 - std::basic_ostringstream, 2124
 - std::basic_stringstream, 2294
 - terminate, 36
 - terminate_handler, 34
 - throw_with_nested, 36
 - uncaught_exception, 36
 - unexpected, 36
 - unexpected_handler, 34
- exp
 - complex_numbers, 44
- expint
 - tr1_math_spec_func, 121
- exponential_distribution
 - std::exponential_distribution, 2598
- extended
 - std::regex_constants, 749
- Extensions, 11
- extptr_allocator.h, 3617
- fabs
 - complex_numbers, 45
- facet
 - std::locale::facet, 2859
- fail
 - std::basic_fstream, 1692
 - std::basic_ifstream, 1761
 - std::basic_ios, 1814
 - std::basic_iostream, 1850
 - std::basic_istream, 1917
 - std::basic_istream, 1975
 - std::basic_ofstream, 2031
 - std::basic_ostream, 2076
 - std::basic_ostringstream, 2125
 - std::basic_stringstream, 2295

- failbit
 - std::basic_fstream, 1741
 - std::basic_ifstream, 1797
 - std::basic_ios, 1829
 - std::basic_iostream, 1898
 - std::basic_istream, 1952
 - std::basic_istream, 2011
 - std::basic_ofstream, 2058
 - std::basic_ostream, 2103
 - std::basic_ostringstream, 2152
 - std::basic_stringstream, 2343
 - std::ios_base, 2750
- failed
 - std::ostreambuf_iterator, 3117
- false_type
 - metaprogramming, 166
- false_name
 - std::numpunct, 3098
 - std::numpunct_byname, 3105
- fd
 - __gnu_cxx::stdio_filebuf, 973
- features.h, 3618
 - _GLIBCXX_BAL_QUICKSORT, 3618
 - _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, 3618
 - _GLIBCXX_FIND_EQUAL_SPLIT, 3618
 - _GLIBCXX_FIND_GROWING_BLOCKS, 3619
 - _GLIBCXX_MERGESORT, 3619
 - _GLIBCXX_QUICKSORT, 3619
 - _GLIBCXX_TREE_DYNAMIC_BALANCING, 3619
 - _GLIBCXX_TREE_FULL_COPY, 3620
 - _GLIBCXX_TREE_INITIAL_SPLITTING, 3620
- fenv.h, 3621
- file
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1004
- filebuf
 - io, 60
- fill
 - mutating_algorithms, 176
 - std::basic_fstream, 1693
 - std::basic_ifstream, 1761, 1762
 - std::basic_ios, 1815
 - std::basic_iostream, 1850
 - std::basic_istream, 1917, 1918
 - std::basic_istream, 1975, 1976
 - std::basic_ofstream, 2031
 - std::basic_ostream, 2077
 - std::basic_ostringstream, 2125
 - std::basic_stringstream, 2295, 2296
- fill_minimal_n
 - __gnu_parallel::_Settings, 1263
- fill_n
 - mutating_algorithms, 177
- find
 - __gnu_cxx::__versa_string, 821, 822
 - __gnu_debug::basic_string, 1094, 1095
 - non_mutating_algorithms, 200
 - std::basic_string, 2219, 2220
 - std::map, 2888, 2889
 - std::multimap, 2980
 - std::multiset, 3000
 - std::set, 3238
- find.h, 3622
- find_end
 - non_mutating_algorithms, 201
- find_first_not_of
 - __gnu_cxx::__versa_string, 823, 824
 - __gnu_debug::basic_string, 1095–1097
 - std::basic_string, 2221, 2222
- find_first_of
 - __gnu_cxx::__versa_string, 825, 826
 - __gnu_debug::basic_string, 1097, 1098
 - non_mutating_algorithms, 202, 203
 - std::basic_string, 2223, 2224
- find_if
 - non_mutating_algorithms, 203
- find_if_not
 - non_mutating_algorithms, 204

-
- find_increasing_factor
 - [__gnu_parallel::_Settings](#), 1263
 - find_initial_block_size
 - [__gnu_parallel::_Settings](#), 1263
 - find_last_not_of
 - [__gnu_cxx::__versa_string](#), 827–829
 - [__gnu_debug::basic_string](#), 1099, 1100
 - [std::basic_string](#), 2224–2226
 - find_last_of
 - [__gnu_cxx::__versa_string](#), 829–831
 - [__gnu_debug::basic_string](#), 1100–1102
 - [std::basic_string](#), 2226–2228
 - find_maximum_block_size
 - [__gnu_parallel::_Settings](#), 1264
 - find_selectors.h, 3623
 - find_sequential_search_size
 - [__gnu_parallel::_Settings](#), 1264
 - first
 - [__gnu_parallel::_IteratorPair](#), 1193
 - [std::pair](#), 3129
 - [std::sub_match](#), 3283
 - first_argument_type
 - [__gnu_cxx::project1st](#), 938
 - [__gnu_cxx::project2nd](#), 940
 - [__gnu_parallel::_EqualFromLess](#), 1186
 - [__gnu_parallel::_EqualTo](#), 1188
 - [__gnu_parallel::_Less](#), 1200
 - [__gnu_parallel::_Lexicographic](#), 1202
 - [__gnu_parallel::_LexicographicReverse](#), 1204
 - [__gnu_parallel::_Multiplies](#), 1240
 - [__gnu_parallel::_Plus](#), 1244
 - [std::_Maybe_unary_or_binary_function<_Res, _T1, _T2 >](#), 1558
 - [std::binary_function](#), 2354
 - [std::binary_negate](#), 2355
 - [std::const_mem_fun1_ref_t](#), 2452
 - [std::const_mem_fun1_t](#), 2454
 - [std::divides](#), 2584
 - [std::equal_to](#), 2590
 - [std::greater](#), 2671
 - [std::greater_equal](#), 2673
 - [std::less](#), 2815
 - [std::less_equal](#), 2817
 - [std::logical_and](#), 2866
 - [std::logical_or](#), 2870
 - [std::mem_fun1_ref_t](#), 2911
 - [std::mem_fun1_t](#), 2913
 - [std::minus](#), 2927
 - [std::modulus](#), 2929
 - [std::multiplies](#), 2990
 - [std::not_equal_to](#), 3026
 - [std::plus](#), 3143
 - [std::pointer_to_binary_function](#), 3145
 - first_type
 - [__gnu_parallel::_IteratorPair](#), 1193
 - [std::pair](#), 3128
 - [std::sub_match](#), 3280
 - fixed
 - [std](#), 620
 - [std::basic_fstream](#), 1741
 - [std::basic_ifstream](#), 1798
 - [std::basic_ios](#), 1829
 - [std::basic_iostream](#), 1898
 - [std::basic_istream](#), 1953
 - [std::basic_istreamstream](#), 2011
 - [std::basic_ofstream](#), 2058
 - [std::basic_ostream](#), 2104
 - [std::basic_ostreamstream](#), 2152
 - [std::basic_stringstream](#), 2343
 - [std::ios_base](#), 2750
 - flags
 - [std::basic_fstream](#), 1693, 1694
 - [std::basic_ifstream](#), 1762
 - [std::basic_ios](#), 1815, 1816
 - [std::basic_iostream](#), 1851
 - [std::basic_istream](#), 1918
 - [std::basic_istreamstream](#), 1976
 - [std::basic_ofstream](#), 2032
 - [std::basic_ostream](#), 2078
 - [std::basic_ostreamstream](#), 2126
 - [std::basic_regex](#), 2164
 - [std::basic_stringstream](#), 2296
 - [std::ios_base](#), 2742
-

- flip
 - std::bitset, 2374
- float_denorm_style
 - std, 600
- float_round_style
 - std, 600
- floatfield
 - std::basic_fstream, 1742
 - std::basic_ifstream, 1798
 - std::basic_ios, 1829
 - std::basic_iostream, 1898
 - std::basic_istream, 1953
 - std::basic_istream, 2011
 - std::basic_ofstream, 2059
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2152
 - std::basic_stringstream, 2344
 - std::ios_base, 2750
- flush
 - std, 620
 - std::basic_fstream, 1694
 - std::basic_iostream, 1851
 - std::basic_ofstream, 2032
 - std::basic_ostream, 2078
 - std::basic_ostringstream, 2126
 - std::basic_stringstream, 2297
- fmtflags
 - std::basic_fstream, 1684
 - std::basic_ifstream, 1753
 - std::basic_ios, 1807
 - std::basic_iostream, 1843
 - std::basic_istream, 1910
 - std::basic_istream, 1967
 - std::basic_ofstream, 2023
 - std::basic_ostream, 2069
 - std::basic_ostringstream, 2117
 - std::basic_stringstream, 2287
 - std::ios_base, 2739
- for_each
 - non_mutating_algorithms, 204
- for_each.h, 3624
- for_each_minimal_n
 - __gnu_parallel::_Settings, 1264
- for_each_selectors.h, 3625
- format
 - std::match_results, 2906
- format_default
 - std::regex_constants, 749
- format_first_only
 - std::regex_constants, 750
- format_no_copy
 - std::regex_constants, 750
- format_sed
 - std::regex_constants, 750
- formatter.h, 3627
- forward
 - std, 620
- forward_list
 - std::forward_list, 2615–2618
- forward_list.h, 3629
- forward_list.tcc, 3631
- fpos
 - std::fpos, 2634
- frac_digits
 - std::moneypunct, 2952
 - std::moneypunct_byname, 2964
- front
 - __gnu_cxx::__versa_string, 831, 832
 - std::deque, 2561
 - std::forward_list, 2624
 - std::list, 2837
 - std::queue, 3163
 - std::vector, 3380, 3381
- front_insert_iterator
 - std::front_insert_iterator, 2639
- front_inserter
 - iterators, 283
- fstream, 3632
 - io, 60
- fstream.tcc, 3633
- functexcept.h, 3634
- function
 - std::function<_Res(_ArgTypes...)>, 2643, 2644
- Function Objects, 262
- functional, 3635, 3640
- functional_hash.h, 3642
- functions.h, 3643
- functors
 - mem_fn, 263
- future, 3646

- std::future, 2650
- std::future< _Res & >, 2653
- std::future< void >, 2656
- future_category
 - futures, 56
- future_errc
 - futures, 56
- Futures, 54
- futures
 - future_category, 56
 - future_errc, 56
- gamma_distribution
 - std::gamma_distribution, 2661
- gbump
 - __gnu_cxx::enc_filebuf, 893
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1004
 - std::basic_filebuf, 1651
 - std::basic_streambuf, 2175
 - std::basic_stringbuf, 2259
- gcount
 - std::basic_fstream, 1695
 - std::basic_ifstream, 1763
 - std::basic_iostream, 1852
 - std::basic_istream, 1919
 - std::basic_istreamstream, 1977
 - std::basic_stringstream, 2297
- generate
 - mutating_algorithms, 177
- generate_canonical
 - random, 261
- generate_minimal_n
 - __gnu_parallel::_Settings, 1264
- generate_n
 - mutating_algorithms, 178
- get
 - std::auto_ptr, 1629
 - std::basic_fstream, 1695–1698
 - std::basic_ifstream, 1763–1766
 - std::basic_iostream, 1852–1855
 - std::basic_istream, 1919–1922
 - std::basic_istreamstream, 1977–1980
 - std::basic_stringstream, 2297–2300
 - std::future, 2650
 - std::future< _Res & >, 2653
 - std::future< void >, 2656
 - std::money_get, 2935, 2936
 - std::num_get, 3033–3037
 - std::shared_future, 3247
 - std::shared_future< _Res & >, 3250
- get_allocator
 - __gnu_cxx::__versa_string, 832
 - __gnu_debug::basic_string, 1102
 - std::basic_string, 2228
 - std::deque, 2562
 - std::forward_list, 2624
 - std::list, 2837
 - std::map, 2889
 - std::multimap, 2981
 - std::multiset, 3001
 - std::set, 3238
- get_date
 - std::time_get, 3297
 - std::time_get_byname, 3307
- get_deleter
 - pointer_abstractions, 67
- get_id
 - std::this_thread, 756
- get_money
 - std, 620
- get_monthname
 - std::time_get, 3297
 - std::time_get_byname, 3308
- get_temporary_buffer
 - std, 621
- get_time
 - std::time_get, 3298
 - std::time_get_byname, 3308
- get_weekday
 - std::time_get, 3299
 - std::time_get_byname, 3309
- get_year
 - std::time_get, 3299
 - std::time_get_byname, 3310
- getline
 - std, 621–623
 - std::basic_fstream, 1698, 1699
 - std::basic_ifstream, 1766, 1767
 - std::basic_iostream, 1855, 1856
 - std::basic_istream, 1922, 1923

- std::basic_istream, 1980, 1981
- std::basic_stringstream, 2301
- getloc
 - __gnu_cxx::enc_filebuf, 894
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1004
 - std::basic_filebuf, 1651
 - std::basic_fstream, 1700
 - std::basic_ifstream, 1768
 - std::basic_ios, 1816
 - std::basic_iostream, 1857
 - std::basic_istream, 1924
 - std::basic_istreamstream, 1982
 - std::basic_ofstream, 2033
 - std::basic_ostream, 2079
 - std::basic_ostringstream, 2127
 - std::basic_regex, 2165
 - std::basic_streambuf, 2175
 - std::basic_stringbuf, 2259
 - std::basic_stringstream, 2302
 - std::ios_base, 2742
 - std::regex_traits, 3202
- global
 - std::locale, 2853
- good
 - std::basic_fstream, 1700
 - std::basic_ifstream, 1768
 - std::basic_ios, 1816
 - std::basic_iostream, 1857
 - std::basic_istream, 1924
 - std::basic_istreamstream, 1982
 - std::basic_ofstream, 2033
 - std::basic_ostream, 2079
 - std::basic_ostringstream, 2127
 - std::basic_stringstream, 2302
- goodbit
 - std::basic_fstream, 1742
 - std::basic_ifstream, 1798
 - std::basic_ios, 1829
 - std::basic_iostream, 1898
 - std::basic_istream, 1953
 - std::basic_istreamstream, 2011
 - std::basic_ofstream, 2059
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2152
- std::basic_stringstream, 2344
- std::ios_base, 2751
- gptr
 - __gnu_cxx::enc_filebuf, 894
 - __gnu_cxx::stdio_filebuf, 975
 - __gnu_cxx::stdio_sync_filebuf, 1005
 - std::basic_filebuf, 1652
 - std::basic_streambuf, 2176
 - std::basic_stringbuf, 2259
- grep
 - std::regex_constants, 750
- grouping
 - std::moneypunct, 2952
 - std::moneypunct_byname, 2964
 - std::numpunct, 3099
 - std::numpunct_byname, 3105
- gslice
 - numeric_arrays, 89
- gslice.h, 3649
- gslice_array
 - numeric_arrays, 89
- gslice_array.h, 3650
- has_denorm
 - std::__numeric_limits_base, 1486
 - std::numeric_limits, 3052
- has_denorm_loss
 - std::__numeric_limits_base, 1486
 - std::numeric_limits, 3052
- has_facet
 - std, 624
 - std::locale::id, 2861
- has_infinity
 - std::__numeric_limits_base, 1486
 - std::numeric_limits, 3052
- has_quiet_NaN
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
- has_signaling_NaN
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
- hash
 - std::collate, 2437
 - std::collate_byname, 2443
- hash_fun.h, 3651

- hash_map, 3652
- hash_policy.hpp, 3654
- hash_set, 3655
- Hashes, 257
- hashtable.h, 3657, 3658
- hashtable_policy.h, 3659
- Heap Algorithms, 273
- heap_algorithms
 - is_heap, 274
 - is_heap_until, 274, 275
 - make_heap, 275, 276
 - pop_heap, 276, 277
 - push_heap, 277
 - sort_heap, 278
- hermite
 - tr1_math_spec_func, 121
- hex
 - std, 624
 - std::basic_fstream, 1742
 - std::basic_ifstream, 1798
 - std::basic_ios, 1830
 - std::basic_iostream, 1899
 - std::basic_istream, 1953
 - std::basic_istreamstream, 2012
 - std::basic_ofstream, 2059
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2153
 - std::basic_stringstream, 2344
 - std::ios_base, 2751
- hours
 - std::chrono, 730
- hyperg
 - tr1_math_spec_func, 121
- I/O, 57
- icase
 - std::regex_constants, 751
- id
 - std::collate, 2438
 - std::collate_byname, 2444
 - std::ctype, 2473
 - std::ctype< char >, 2486
 - std::ctype< wchar_t >, 2502
 - std::ctype_byname, 2518
 - std::ctype_byname< char >, 2531
 - std::locale::id, 2860
 - std::messages, 2921
 - std::messages_byname, 2925
 - std::money_get, 2937
 - std::money_put, 2943
 - std::moneypunct, 2955
 - std::moneypunct_byname, 2967
 - std::num_get, 3038
 - std::num_put, 3048
 - std::numpunct, 3100
 - std::numpunct_byname, 3107
 - std::time_get, 3300
 - std::time_get_byname, 3310
 - std::time_put, 3316
 - std::time_put_byname, 3321
- identity_element
 - SGIextensions, 19, 20
- ifstream
 - io, 60
- ignore
 - std::basic_fstream, 1700, 1701
 - std::basic_ifstream, 1768, 1769
 - std::basic_iostream, 1857, 1858
 - std::basic_istream, 1924, 1925
 - std::basic_istreamstream, 1982, 1983
 - std::basic_stringstream, 2303, 2304
- imbue
 - __gnu_cxx::enc_filebuf, 894
 - __gnu_cxx::stdio_filebuf, 975
 - __gnu_cxx::stdio_sync_filebuf, 1005
 - std::basic_filebuf, 1652
 - std::basic_fstream, 1702
 - std::basic_ifstream, 1770
 - std::basic_ios, 1817
 - std::basic_iostream, 1859
 - std::basic_istream, 1926
 - std::basic_istreamstream, 1984
 - std::basic_ofstream, 2033
 - std::basic_ostream, 2079
 - std::basic_ostringstream, 2127
 - std::basic_regex, 2165
 - std::basic_streambuf, 2176
 - std::basic_stringbuf, 2260
 - std::basic_stringstream, 2304
 - std::ios_base, 2743
 - std::regex_traits, 3202

- in
- std::__codecvt_abstract_base, 1388
 - std::basic_fstream, 1742
 - std::basic_ifstream, 1798
 - std::basic_ios, 1830
 - std::basic_iostream, 1899
 - std::basic_istream, 1954
 - std::basic_istream, 2012
 - std::basic_ofstream, 2059
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2153
 - std::basic_stringstream, 2344
 - std::codecvt, 2403
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2409
 - std::codecvt< char, char, mbstate_t >, 2415
 - std::codecvt< wchar_t, char, mbstate_t >, 2421
 - std::codecvt_byname, 2428
 - std::ios_base, 2751
- in_avail
- __gnu_cxx::enc_filebuf, 895
 - __gnu_cxx::stdio_filebuf, 976
 - __gnu_cxx::stdio_sync_filebuf, 1006
 - std::basic_filebuf, 1653
 - std::basic_streambuf, 2177
 - std::basic_stringbuf, 2260
- include/ Directory Reference, 322
- include/backward/ Directory Reference, 307
- include/bits/ Directory Reference, 310
- include/debug/ Directory Reference, 313
- include/decimal/ Directory Reference, 314
- include/ext/ Directory Reference, 318
- include/ext/pb_ds/ Directory Reference, 329
- include/ext/pb_ds/detail/ Directory Reference, 315
- include/parallel/ Directory Reference, 327
- include/profile/ Directory Reference, 330
- include/profile/impl/ Directory Reference, 321
- include/tr1/ Directory Reference, 332
- include/tr1/impl/ Directory Reference, 333
- include/x86_64-unknown-linux-gnu/ Directory Reference, 334
- include/x86_64-unknown-linux-gnu/bits/ Directory Reference, 309
- includes
- set_algorithms, 233
- increment
- std::linear_congruential_engine, 2824
- independent_bits_engine
- std::independent_bits_engine, 2720, 2721
- indirect_array
- numeric_arrays, 89
- indirect_array.h, 3661
- infinity
- std::numeric_limits, 3050
- init
- std::basic_fstream, 1702
 - std::basic_ifstream, 1770
 - std::basic_ios, 1817
 - std::basic_iostream, 1860
 - std::basic_istream, 1926
 - std::basic_istream, 1984
 - std::basic_ofstream, 2034
 - std::basic_ostream, 2080
 - std::basic_ostringstream, 2128
 - std::basic_stringstream, 2305
- initializer_list, 3662
- inner_product
- std, 624, 625
- inplace_merge
- sorting_algorithms, 213
- insert
- __gnu_cxx::__versa_string, 832–837
 - __gnu_debug::basic_string, 1102–1107
 - std::basic_string, 2229–2233
 - std::deque, 2562, 2563
 - std::list, 2837–2839
 - std::map, 2890, 2891
 - std::multimap, 2981, 2982
 - std::multiset, 3001, 3002

- std::set, 3238–3240
- std::vector, 3381, 3382
- insert_after
 - std::forward_list, 2624–2626
- insert_iterator
 - std::insert_iterator, 2731
- inserter
 - iterators, 284
- int_type
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 1002
 - std::basic_filebuf, 1647
 - std::basic_fstream, 1685
 - std::basic_ifstream, 1754
 - std::basic_ios, 1808
 - std::basic_iostream, 1843
 - std::basic_istream, 1911
 - std::basic_istreamstream, 1968
 - std::basic_ofstream, 2024
 - std::basic_ostream, 2070
 - std::basic_ostringstream, 2118
 - std::basic_streambuf, 2172
 - std::basic_stringbuf, 2255
 - std::basic_stringstream, 2288
 - std::istreambuf_iterator, 2803
- internal
 - std, 625
 - std::basic_fstream, 1743
 - std::basic_ifstream, 1799
 - std::basic_ios, 1830
 - std::basic_iostream, 1899
 - std::basic_istream, 1954
 - std::basic_istreamstream, 2012
 - std::basic_ofstream, 2060
 - std::basic_ostream, 2105
 - std::basic_ostringstream, 2153
 - std::basic_stringstream, 2345
 - std::ios_base, 2751
- intervals
 - std::piecewise_constant_-distribution, 3131
 - std::piecewise_linear_distribution, 3137
- intl
- std::moneypunct, 2955
- std::moneypunct_byname, 2967
- io
 - filebuf, 60
 - fstream, 60
 - ifstream, 60
 - ios, 60
 - iostream, 60
 - istream, 60
 - istreamstream, 61
 - ofstream, 61
 - ostream, 61
 - ostringstream, 61
 - streambuf, 61
 - stringbuf, 61
 - stringstream, 61
 - wfilebuf, 62
 - wfstream, 62
 - wifstream, 62
 - wios, 62
 - wiostream, 62
 - wistream, 62
 - wstringstream, 62
 - wofstream, 63
 - wostream, 63
 - wstringstream, 63
- iomanip, 3663
- ios, 3665
 - io, 60
- ios_base.h, 3666
- iosfwd, 3669
- iostate
 - std::basic_fstream, 1685
 - std::basic_ifstream, 1754
 - std::basic_ios, 1808
 - std::basic_iostream, 1844
 - std::basic_istream, 1911
 - std::basic_istreamstream, 1968
 - std::basic_ofstream, 2024
 - std::basic_ostream, 2070
 - std::basic_ostringstream, 2118
 - std::basic_stringstream, 2288
 - std::ios_base, 2740

-
- iostream, 3670
 - io, 60
 - iota
 - SGIextensions, 20
 - std, 625
 - is
 - std::__ctype_abstract_base, 1400, 1401
 - std::ctype, 2467, 2468
 - std::ctype< char >, 2481
 - std::ctype< wchar_t >, 2497, 2498
 - std::ctype_byname, 2512, 2513
 - std::ctype_byname< char >, 2525, 2526
 - is_bounded
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
 - is_exact
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
 - is_heap
 - heap_algorithms, 274
 - SGIextensions, 20
 - is_heap_until
 - heap_algorithms, 274, 275
 - is_iec559
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
 - is_integer
 - std::__numeric_limits_base, 1487
 - std::numeric_limits, 3053
 - is_modulo
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3054
 - is_open
 - __gnu_cxx::enc_filebuf, 895
 - __gnu_cxx::stdio_filebuf, 976
 - std::basic_filebuf, 1653
 - std::basic_fstream, 1703
 - std::basic_ifstream, 1771
 - std::basic_ofstream, 2034
 - is_partitioned
 - mutating_algorithms, 178
 - is_signed
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3054
 - is_sorted
 - SGIextensions, 21
 - sorting_algorithms, 214
 - is_sorted_until
 - sorting_algorithms, 215
 - is_specialized
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3054
 - isalnum
 - std, 626
 - isalpha
 - std, 626
 - isctrl
 - std, 626
 - isctype
 - tr1_regex, 132
 - isdigit
 - std, 626
 - isgraph
 - std, 626
 - islower
 - std, 626
 - isprint
 - std, 627
 - ispunct
 - std, 627
 - isspace
 - std, 627
 - istream, 3671
 - io, 60
 - istream.tcc, 3673
 - istream_iterator
 - std::istream_iterator, 2801
 - istream_type
 - std::istreambuf_iterator, 2803
 - istreambuf_iterator
 - std::istreambuf_iterator, 2805
 - istringstream
 - io, 61
 - isupper
 - std, 627
 - isxdigit
 - std, 627
 - iter_swap
 - mutating_algorithms, 179
 - iter_type
-

- std::money_get, 2933
- std::money_put, 2939
- std::num_get, 3029
- std::num_put, 3041
- std::time_get, 3292
- std::time_get_byname, 3303
- std::time_put, 3313
- std::time_put_byname, 3318
- iterator, 3674, 3675
 - std::set, 3229
- iterator.h, 3676
- iterator_category
 - std::back_insert_iterator, 1634
 - std::front_insert_iterator, 2638
 - std::insert_iterator, 2730
 - std::istream_iterator, 2800
 - std::istreambuf_iterator, 2804
 - std::iterator, 2808
 - std::ostream_iterator, 3110
 - std::ostreambuf_iterator, 3115
 - std::raw_storage_iterator, 3182
 - std::reverse_iterator, 3215
- iterator_tracker.h, 3677
- Iterators, 279
- iterators
 - __iterator_category, 283
 - back_inserter, 283
 - front_inserter, 283
 - inserter, 284
 - operator==, 284, 285
- iword
 - std::basic_fstream, 1703
 - std::basic_ifstream, 1771
 - std::basic_ios, 1817
 - std::basic_istream, 1860
 - std::basic_istream, 1927
 - std::basic_istreamstream, 1985
 - std::basic_ofstream, 2034
 - std::basic_ostream, 2080
 - std::basic_ostreamstream, 2128
 - std::basic_stringstream, 2305
 - std::ios_base, 2743
- k
 - std::negative_binomial_distribution, 3014
- key_comp
 - std::map, 2891
 - std::multimap, 2983
 - std::multiset, 3003
 - std::set, 3240
- key_compare
 - std::set, 3229
- key_type
 - std::set, 3230
- kill_dependency
 - atomics, 256
- L1_cache_size
 - __gnu_parallel::_Settings, 1264
- L2_cache_size
 - __gnu_parallel::_Settings, 1264
- laguerre
 - tr1_math_spec_func, 122
- lambda
 - std::exponential_distribution, 2598
- left
 - std, 627
 - std::basic_fstream, 1743
 - std::basic_ifstream, 1799
 - std::basic_ios, 1831
 - std::basic_istream, 1900
 - std::basic_istream, 1954
 - std::basic_istreamstream, 2013
 - std::basic_ofstream, 2060
 - std::basic_ostream, 2105
 - std::basic_ostreamstream, 2154
 - std::basic_stringstream, 2345
 - std::ios_base, 2752
- legendre
 - tr1_math_spec_func, 122
- length
 - __gnu_cxx::_versa_string, 837
 - __gnu_debug::basic_string, 1107
 - std::basic_string, 2233
 - std::match_results, 2906
 - std::regex_traits, 3203
 - std::sub_match, 3282
- lexicographical_compare
 - sorting_algorithms, 215, 216
- lexicographical_compare_3way
 - SGlextensions, 21

- limits, [3679](#)
- linear_congruential_engine
 - std::linear_congruential_engine, [2819](#), [2820](#)
- list, [3682](#), [3683](#), [3685](#)
 - std::list, [2829–2831](#)
- list.tcc, [3687](#)
- list_partition
 - __gnu_parallel, [434](#)
- list_partition.h, [3688](#)
- list_update_policy.hpp, [3689](#)
- locale, [3690](#)
 - std::locale, [2851](#), [2852](#)
- locale_classes.h, [3691](#)
- locale_classes.tcc, [3693](#)
- locale_facets.h, [3694](#)
- locale_facets.tcc, [3697](#)
- locale_facets_nonio.h, [3699](#)
- locale_facets_nonio.tcc, [3701](#)
- localefwd.h, [3703](#)
- Locales, [258](#)
- lock
 - mutexes, [70](#)
- log
 - complex_numbers, [45](#)
- log10
 - complex_numbers, [45](#)
- logic_error
 - std::logic_error, [2863](#)
- lookup_classname
 - std::regex_traits, [3203](#)
- lookup_collatename
 - std::regex_traits, [3204](#)
- losertree.h, [3705](#)
- lower_bound
 - binary_search_algorithms, [243](#), [244](#)
 - std::map, [2892](#)
 - std::multimap, [2983](#), [2984](#)
 - std::multiset, [3003](#), [3004](#)
 - std::set, [3240](#), [3241](#)
- lowest
 - std::numeric_limits, [3051](#)
- macros.h, [3707](#)
 - _GLIBCXX_DEBUG_VERIFY, [3709](#)
 - __glibcxx_check_erase, [3707](#)
 - __glibcxx_check_erase_range, [3707](#)
 - __glibcxx_check_heap_pred, [3708](#)
 - __glibcxx_check_insert, [3708](#)
 - __glibcxx_check_insert_range, [3708](#)
 - __glibcxx_check_partitioned_lower, [3708](#)
 - __glibcxx_check_partitioned_lower_pred, [3708](#)
 - __glibcxx_check_partitioned_upper_pred, [3709](#)
 - __glibcxx_check_sorted_pred, [3709](#)
- make_heap
 - heap_algorithms, [275](#), [276](#)
- make_shared
 - pointer_abstractions, [67](#)
- malloc_allocator.h, [3710](#)
- map, [3711–3713](#)
 - std::map, [2880–2882](#)
- map.h, [3714](#), [3716](#)
- mark_count
 - std::basic_regex, [2165](#)
- mask_array
 - numeric_arrays, [89](#)
- mask_array.h, [3718](#)
- match_any
 - std::regex_constants, [751](#)
- match_continuous
 - std::regex_constants, [751](#)
- match_default
 - std::regex_constants, [751](#)
- match_flag_type
 - std::regex_constants, [745](#)
- match_not_bol
 - std::regex_constants, [751](#)
- match_not_bow
 - std::regex_constants, [751](#)
- match_not_eol
 - std::regex_constants, [752](#)
- match_not_eow
 - std::regex_constants, [752](#)
- match_not_null
 - std::regex_constants, [752](#)
- match_prev_avail
 - std::regex_constants, [752](#)
- match_results

- std::match_results, 2904
- Mathematical Special Functions, 116
- max
 - __gnu_parallel, 435
 - numeric_arrays, 93
 - sorting_algorithms, 216, 217
 - std::bernoulli_distribution, 2349
 - std::binomial_distribution, 2362
 - std::cauchy_distribution, 2382
 - std::chi_squared_distribution, 2391
 - std::discard_block_engine, 2574
 - std::discrete_distribution, 2579
 - std::exponential_distribution, 2598
 - std::extreme_value_distribution, 2603
 - std::fisher_f_distribution, 2607
 - std::gamma_distribution, 2662
 - std::geometric_distribution, 2667
 - std::independent_bits_engine, 2722
 - std::linear_congruential_engine, 2820
 - std::lognormal_distribution, 2872
 - std::negative_binomial_distribution, 3014
 - std::normal_distribution, 3021
 - std::numeric_limits, 3051
 - std::piecewise_constant_distribution, 3132
 - std::piecewise_linear_distribution, 3138
 - std::poisson_distribution, 3149
 - std::shuffle_order_engine, 3263
 - std::student_t_distribution, 3275
 - std::uniform_int_distribution, 3341
 - std::uniform_real_distribution, 3346
 - std::weibull_distribution, 3396
- max_digits10
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3054
- max_element
 - sorting_algorithms, 217, 218
- max_element_minimal_n
 - __gnu_parallel::_Settings, 1265
- max_exponent
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3054
- max_exponent10
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3055
- max_size
 - __gnu_cxx::__versa_string, 838
 - __gnu_debug::basic_string, 1107
 - std::basic_string, 2234
 - std::deque, 2564
 - std::forward_list, 2626
 - std::list, 2839
 - std::map, 2892
 - std::multimap, 2984
 - std::multiset, 3004
 - std::set, 3241
 - std::vector, 3383
- mean
 - std::normal_distribution, 3021
 - std::poisson_distribution, 3149
- mem_fn
 - functors, 263
- Memory, 64
- memory, 3719, 3720
- memory_order
 - atomics, 256
- merge
 - sorting_algorithms, 218, 219
 - std::forward_list, 2626, 2627
 - std::list, 2840
- merge.h, 3722
- merge_minimal_n
 - __gnu_parallel::_Settings, 1265
- merge_oversampling
 - __gnu_parallel::_Settings, 1265
- messages
 - std::locale, 2856
 - std::messages, 2920, 2921
- messages_members.h, 3724
- metaprogramming
 - false_type, 166
 - true_type, 166
- microseconds
 - std::chrono, 730
- milliseconds
 - std::chrono, 730
- min
 - __gnu_parallel, 435

- numeric_arrays, 93
- sorting_algorithms, 219, 220
- std::bernoulli_distribution, 2349
- std::binomial_distribution, 2362
- std::cauchy_distribution, 2382
- std::chi_squared_distribution, 2391
- std::discard_block_engine, 2575
- std::discrete_distribution, 2579
- std::exponential_distribution, 2598
- std::extreme_value_distribution, 2603
- std::fisher_f_distribution, 2607
- std::gamma_distribution, 2662
- std::geometric_distribution, 2667
- std::independent_bits_engine, 2722
- std::linear_congruential_engine, 2821
- std::lognormal_distribution, 2872
- std::negative_binomial_distribution, 3014
- std::normal_distribution, 3021
- std::numeric_limits, 3051
- std::piecewise_constant_distribution, 3132
- std::piecewise_linear_distribution, 3138
- std::poisson_distribution, 3149
- std::shuffle_order_engine, 3264
- std::student_t_distribution, 3275
- std::uniform_int_distribution, 3341
- std::uniform_real_distribution, 3346
- std::weibull_distribution, 3396
- min_element
 - sorting_algorithms, 221
- min_element_minimal_n
 - __gnu_parallel::_Settings, 1265
- min_exponent
 - std::__numeric_limits_base, 1488
 - std::numeric_limits, 3055
- min_exponent10
 - std::__numeric_limits_base, 1489
 - std::numeric_limits, 3055
- minmax
 - sorting_algorithms, 221, 222
- minmax_element
 - sorting_algorithms, 222
- minstd_rand
 - random_generators, 289
- minstd_rand0
 - random_generators, 289
- minutes
 - std::chrono, 730
- mismatch
 - non_mutating_algorithms, 204, 205
- modulus
 - std::linear_congruential_engine, 2824
- monetary
 - std::locale, 2856
- money_get
 - std::money_get, 2934
- money_put
 - std::money_put, 2940
- money_punct
 - std::money_punct, 2947
- move
 - mutating_algorithms, 179, 180
- move.h, 3725
- move_backward
 - mutating_algorithms, 180
- mt19937
 - random_generators, 289
- mt19937_64
 - random_generators, 290
- mt_allocator.h, 3727
- multimap
 - std::multimap, 2972–2974
- multimap.h, 3729, 3731
- multiplier
 - std::linear_congruential_engine, 2824
- multiseq_partition
 - __gnu_parallel, 435
- multiseq_selection
 - __gnu_parallel, 436
- multiseq_selection.h, 3733
- multiset
 - std::multiset, 2993–2995
- multiset.h, 3735, 3737
- multiway_merge
 - __gnu_parallel, 437
- multiway_merge.h, 3739

- neg_format
 - std::moneypunct, 2953
 - std::moneypunct_byname, 2964
- negative_sign
 - std::moneypunct, 2953
 - std::moneypunct_byname, 2965
- Negators, 267
- negators
 - not1, 268
 - not2, 268
- nested_exception.h, 3749
- new, 3750
 - operator delete, 3751
 - operator new, 3753
- new_allocator.h, 3756
- new_handler
 - std, 598
- next_permutation
 - sorting_algorithms, 223
- nboolalpha
 - std, 627
- Non-Mutating Algorithms, 195
- non_mutating_algorithms
 - adjacent_find, 197
 - all_of, 197
 - any_of, 198
 - count, 198
 - count_if, 199
 - equal, 199, 200
 - find, 200
 - find_end, 201
 - find_first_of, 202, 203
 - find_if, 203
 - find_if_not, 204
 - for_each, 204
 - mismatch, 204, 205
 - none_of, 205
 - search, 206, 207
 - search_n, 207, 208
- none
 - std::bitset, 2374
 - std::locale, 2856
- none_of
 - non_mutating_algorithms, 205
- norm
 - complex_numbers, 45
- Normal Distributions, 295
- normal_distribution
 - std::normal_distribution, 3020
- noshowbase
 - std, 628
- noshowpoint
 - std, 628
- noshowpos
 - std, 628
- noskipws
 - std, 628
- nosubs
 - std::regex_constants, 752
- not1
 - negators, 268
- not2
 - negators, 268
- nounitbuf
 - std, 628
- nouppercase
 - std, 628
- npos
 - __gnu_cxx::__versa_string, 856
 - __gnu_debug::basic_string, 1124
 - std::basic_string, 2251
- nth_element
 - sorting_algorithms, 224, 225
- nth_element_minimal_n
 - __gnu_parallel::_Settings, 1266
- num_get
 - std::num_get, 3030
- num_put
 - std::num_put, 3041
- numeric, 3757–3759
 - std::locale, 2856
- Numeric Arrays, 77
- numeric_arrays
 - ~gslice, 92
 - apply, 92
 - cshift, 92
 - gslice, 89
 - gslice_array, 89
 - indirect_array, 89
 - mask_array, 89
 - max, 93
 - min, 93

- operator<<=, 100, 101
- operator>>=, 105, 106
- operator*=, 96
- operator~, 112
- operator^=, 110, 111
- operator+, 97
- operator+=, 97, 98
- operator-, 98
- operator-=, 98, 99
- operator/=, 99, 100
- operator=, 101–105
- operator%=, 93, 94
- operator&=, 95
- resize, 113
- shift, 113
- size, 113, 114
- slice, 90
- slice_array, 90
- start, 114
- stride, 114
- sum, 114
- valarray, 90, 91
- numeric_traits.h, 3763
- numeric_fwd.h, 3764
- Numerics, 72
- num_punct
 - std::num_punct, 3095, 3096
- oct
 - std, 629
 - std::basic_fstream, 1743
 - std::basic_ifstream, 1799
 - std::basic_ios, 1831
 - std::basic_iostream, 1900
 - std::basic_istream, 1954
 - std::basic_istreamstream, 2013
 - std::basic_ofstream, 2060
 - std::basic_ostream, 2105
 - std::basic_ostreamstream, 2154
 - std::basic_stringstream, 2345
 - std::ios_base, 2752
- off_type
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 1002
- std::basic_filebuf, 1647
- std::basic_fstream, 1686
- std::basic_ifstream, 1755
- std::basic_ios, 1809
- std::basic_iostream, 1844
- std::basic_istream, 1912
- std::basic_istreamstream, 1969
- std::basic_ofstream, 2024
- std::basic_ostream, 2071
- std::basic_ostreamstream, 2118
- std::basic_streambuf, 2172
- std::basic_stringbuf, 2256
- std::basic_stringstream, 2289
- ofstream
 - io, 61
- omp_loop.h, 3767
- omp_loop_static.h, 3768
- open
 - __gnu_cxx::enc_filebuf, 895, 896
 - __gnu_cxx::stdio_filebuf, 976, 977
 - std::basic_filebuf, 1653, 1654
 - std::basic_fstream, 1704
 - std::basic_ifstream, 1772
 - std::basic_ofstream, 2035, 2036
- openmode
 - std::basic_fstream, 1686
 - std::basic_ifstream, 1755
 - std::basic_ios, 1809
 - std::basic_iostream, 1844
 - std::basic_istream, 1912
 - std::basic_istreamstream, 1969
 - std::basic_ofstream, 2024
 - std::basic_ostream, 2071
 - std::basic_ostreamstream, 2118
 - std::basic_stringstream, 2289
 - std::ios_base, 2740
- operator_iterator
 - __gnu_debug::_Safe_iterator, 1050
- operator_RAlter
 - __gnu_parallel::_GuardedIterator, 1190
- operator basic_string< value_type >
 - std::sub_match, 3282
- operator bool
 - std::basic_istream::sentry, 1958
 - std::basic_ostream::sentry, 2109

- std::function< _Res(_- ArgTypes...)>, 2644
- operator delete
 - new, 3751
- operator new
 - new, 3753
- operator streamoff
 - std::fpos, 2635
- operator void *
 - std::basic_fstream, 1705
 - std::basic_ifstream, 1773
 - std::basic_ios, 1818
 - std::basic_iostream, 1861
 - std::basic_istream, 1928
 - std::basic_istream, 1986
 - std::basic_ofstream, 2036
 - std::basic_ostream, 2081
 - std::basic_ostringstream, 2129
 - std::basic_stringstream, 2306
- operator<
 - __gnu_cxx, 362
 - __gnu_parallel::_GuardedIterator, 1191
 - std, 635–641
 - std::multiset, 3008
 - tr1_regex, 135–137
- operator<<
 - complex_numbers, 49
 - pointer_abstractions, 68
 - random_distributions_bernoulli, 298
 - random_distributions_normal, 296
 - random_distributions_poisson, 301, 302
 - random_distributions_uniform, 292, 293
 - random_generators, 290
 - std, 641–648
 - std::basic_fstream, 1705–1714
 - std::basic_iostream, 1861–1870
 - std::basic_ofstream, 2037–2046
 - std::basic_ostream, 2081–2090
 - std::basic_ostringstream, 2130–2139
 - std::basic_stringstream, 2306–2315
 - std::binomial_distribution, 2364
 - std::bitset, 2375
 - std::chi_squared_distribution, 2392
 - std::discard_block_engine, 2576
 - std::discrete_distribution, 2580
 - std::fisher_f_distribution, 2608
 - std::gamma_distribution, 2663
 - std::linear_congruential_engine, 2822
 - std::lognormal_distribution, 2873
 - std::negative_binomial_distribution, 3016
 - std::normal_distribution, 3022
 - std::piecewise_constant_distribution, 3133
 - std::piecewise_linear_distribution, 3139
 - std::poisson_distribution, 3151
 - std::shuffle_order_engine, 3265
 - std::student_t_distribution, 3276
 - tr1_regex, 138
- operator<=<=
 - numeric_arrays, 100, 101
 - std::bitset, 2375
- operator<=
 - __gnu_cxx, 363, 364
 - __gnu_parallel::_GuardedIterator, 1191
 - std, 648–651
 - std::rel_ops, 754
 - tr1_regex, 138–140
- operator>
 - __gnu_cxx, 366, 367
 - std, 659–661
 - std::rel_ops, 755
 - tr1_regex, 144–146
- operator>>
 - complex_numbers, 50
 - random_distributions_bernoulli, 298, 299
 - random_distributions_normal, 296
 - random_distributions_poisson, 302, 303
 - random_distributions_uniform, 293, 294
 - std, 665–672
 - std::basic_fstream, 1714–1723
 - std::basic_ifstream, 1773–1782
 - std::basic_iostream, 1871–1880

- std::basic_istream, 1928–1937
- std::basic_istream, 1986–1995
- std::basic_stringstream, 2316–2325
- std::binomial_distribution, 2364
- std::bitset, 2376
- std::chi_squared_distribution, 2392
- std::discard_block_engine, 2577
- std::discrete_distribution, 2580
- std::fisher_f_distribution, 2608
- std::gamma_distribution, 2664
- std::independent_bits_engine, 2724
- std::linear_congruential_engine, 2823
- std::lognormal_distribution, 2873
- std::negative_binomial_distribution, 3016
- std::normal_distribution, 3023
- std::piecewise_constant_distribution, 3133
- std::piecewise_linear_distribution, 3139
- std::poisson_distribution, 3151
- std::shuffle_order_engine, 3266
- std::student_t_distribution, 3276
- operator>>=
 - numeric_arrays, 105, 106
 - std::bitset, 2376
- operator>=
 - __gnu_cxx, 367, 368
 - std, 662–665
 - std::rel_ops, 755
 - tr1_regex, 147–149
- operator*
 - __gnu_debug::_Safe_iterator, 1050
 - __gnu_parallel::_GuardedIterator, 1190
 - complex_numbers, 46
 - std::auto_ptr, 1629
 - std::back_insert_iterator, 1635
 - std::front_insert_iterator, 2639
 - std::insert_iterator, 2731
 - std::istreambuf_iterator, 2805
 - std::ostreambuf_iterator, 3117
 - std::regex_iterator, 3192
 - std::regex_token_iterator, 3198
 - std::reverse_iterator, 3217
- operator*=
 - complex_numbers, 46, 47
 - numeric_arrays, 96
- operator~
 - numeric_arrays, 112
 - std::bitset, 2377
- operator^
 - std, 673
- operator^=
 - numeric_arrays, 110, 111
 - std::bitset, 2377
- operator()
 - __gnu_cxx::subtractive_rng, 1023
 - __gnu_parallel::_Nothing, 1241
 - __gnu_parallel::_RandomNumber, 1255
 - __gnu_parallel::_accumulate_selector, 1126
 - __gnu_parallel::_adjacent_find_selector, 1131
 - __gnu_parallel::_count_if_selector, 1136
 - __gnu_parallel::_count_selector, 1138
 - __gnu_parallel::_fill_selector, 1140
 - __gnu_parallel::_find_first_of_selector, 1143
 - __gnu_parallel::_find_if_selector, 1145
 - __gnu_parallel::_for_each_selector, 1146
 - __gnu_parallel::_generate_selector, 1148
 - __gnu_parallel::_identity_selector, 1153
 - __gnu_parallel::_inner_product_selector, 1156
 - __gnu_parallel::_mismatch_selector, 1161
 - __gnu_parallel::_replace_if_selector, 1169
 - __gnu_parallel::_replace_selector, 1172
 - __gnu_parallel::_transform1_selector, 1173

- __gnu_parallel::__transform2_-
 selector, 1175
- std::bernoulli_distribution, 2349
- std::binomial_distribution, 2362
- std::discard_block_engine, 2575
- std::function< _Res(_-
 ArgTypes...)>, 2645
- std::gamma_distribution, 2662
- std::independent_bits_engine, 2723
- std::linear_congruential_engine,
 2821
- std::locale, 2854
- std::normal_distribution, 3021
- std::poisson_distribution, 3149
- std::shuffle_order_engine, 3264
- std::uniform_int_distribution, 3342
- operator+
 - __gnu_cxx, 359–361
 - complex_numbers, 47
 - numeric_arrays, 97
 - std, 633–635
 - std::fpos, 2635
 - std::reverse_iterator, 3218
- operator++
 - __gnu_debug::_Safe_iterator, 1051
 - __gnu_parallel::_GuardedIterator,
 1190
 - std::back_insert_iterator, 1635
 - std::front_insert_iterator, 2639
 - std::insert_iterator, 2731
 - std::istreambuf_iterator, 2806
 - std::ostreambuf_iterator, 3117
 - std::regex_iterator, 3192
 - std::regex_token_iterator, 3199
 - std::reverse_iterator, 3218
- operator+=
 - __gnu_cxx::__versa_string, 838,
 839
 - __gnu_debug::basic_string, 1107–
 1109
 - complex_numbers, 47
 - numeric_arrays, 97, 98
 - std::basic_string, 2234, 2235
 - std::complex, 2446
 - std::fpos, 2635
 - std::reverse_iterator, 3218
- operator-
 - complex_numbers, 48
 - numeric_arrays, 98
 - std::fpos, 2635
 - std::reverse_iterator, 3219
- operator->
 - __gnu_debug::_Safe_iterator, 1052
 - std::auto_ptr, 1629
 - std::regex_iterator, 3193
 - std::regex_token_iterator, 3199
 - std::reverse_iterator, 3220
- operator--
 - __gnu_debug::_Safe_iterator, 1051,
 1052
 - std::reverse_iterator, 3219
- operator-=
 - complex_numbers, 48
 - numeric_arrays, 98, 99
 - std::complex, 2446
 - std::fpos, 2635
 - std::reverse_iterator, 3220
- operator/
 - complex_numbers, 48, 49
- operator/=
 - complex_numbers, 49
 - numeric_arrays, 99, 100
- operator=
 - __gnu_cxx::__versa_string, 839–
 841
 - __gnu_debug::_Safe_iterator, 1052
 - __gnu_debug::basic_string, 1109,
 1110
 - complex_numbers, 50
 - numeric_arrays, 101–105
 - std::auto_ptr, 1629, 1630
 - std::back_insert_iterator, 1636
 - std::basic_regex, 2165, 2166
 - std::basic_string, 2235–2237
 - std::deque, 2564, 2565
 - std::forward_list, 2627, 2628
 - std::front_insert_iterator, 2640
 - std::function< _Res(_-
 ArgTypes...)>, 2645, 2646
 - std::insert_iterator, 2732
 - std::list, 2840, 2841
 - std::locale, 2854

- std::map, 2893
- std::match_results, 2907
- std::multimap, 2984, 2985
- std::multiset, 3004, 3005
- std::ostream_iterator, 3112
- std::ostreambuf_iterator, 3118
- std::regex_iterator, 3193
- std::regex_token_iterator, 3200
- std::set, 3241, 3242
- std::vector, 3383, 3384
- operator==
 - __gnu_cxx, 364, 365
 - complex_numbers, 50
 - iterators, 284, 285
 - std, 652–658
 - std::bitset, 2375
 - std::discard_block_engine, 2576
 - std::independent_bits_engine, 2724
 - std::linear_congruential_engine, 2822
 - std::locale, 2855
 - std::multiset, 3008
 - std::regex_iterator, 3193
 - std::regex_token_iterator, 3200
 - std::shuffle_order_engine, 3265
 - tr1_regex, 141–144
- operator%=
 - numeric_arrays, 93, 94
- operator&
 - std, 633
- operator&=
 - numeric_arrays, 95
 - std::bitset, 2375
- optimize
 - std::regex_constants, 752
- os_defines.h, 3769
- ostream, 3770
 - io, 61
- ostream.tcc, 3772
- ostream_insert.h, 3773
- ostream_iterator
 - std::ostream_iterator, 3112
- ostream_type
 - std::ostream_iterator, 3110
 - std::ostreambuf_iterator, 3115
- ostreambuf_iterator
 - std::ostreambuf_iterator, 3117
- ostringstream
 - io, 61
- out
 - std::__codecvt_abstract_base, 1389
 - std::basic_fstream, 1743
 - std::basic_ifstream, 1799
 - std::basic_ios, 1831
 - std::basic_iostream, 1900
 - std::basic_istream, 1954
 - std::basic_istream_iterator, 2013
 - std::basic_ofstream, 2060
 - std::basic_ostream, 2105
 - std::basic_ostream_iterator, 2154
 - std::basic_stringstream, 2345
 - std::codecvt, 2404
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2410
 - std::codecvt< char, char, mbstate_t >, 2416
 - std::codecvt< wchar_t, char, mbstate_t >, 2422
 - std::codecvt_byname, 2429
 - std::ios_base, 2752
- overflow
 - __gnu_cxx::enc_filebuf, 896
 - __gnu_cxx::stdio_filebuf, 977
 - __gnu_cxx::stdio_sync_filebuf, 1006
 - std::basic_filebuf, 1654
 - std::basic_streambuf, 2177
 - std::basic_stringbuf, 2261
- p
 - std::bernoulli_distribution, 2349
 - std::binomial_distribution, 2363
 - std::geometric_distribution, 2667
 - std::negative_binomial_distribution, 3014
- pair
 - std::pair, 3128, 3129
- par_loop.h, 3774
- parallel.h, 3775
- parallel_balanced
 - __gnu_parallel, 401
- parallel_omp_loop

- __gnu_parallel, 401
- parallel_omp_loop_static
 - __gnu_parallel, 401
- parallel_taskqueue
 - __gnu_parallel, 402
- parallel_unbalanced
 - __gnu_parallel, 401
- parallel_multiway_merge
 - __gnu_parallel, 445
- parallel_sort_mwms
 - __gnu_parallel, 446
- parallel_sort_mwms_pu
 - __gnu_parallel, 446
- parallel_tag
 - __gnu_parallel::parallel_tag, 1297
- param
 - std::bernoulli_distribution, 2349, 2350
 - std::binomial_distribution, 2363
 - std::cauchy_distribution, 2382
 - std::chi_squared_distribution, 2391, 2392
 - std::discrete_distribution, 2579, 2580
 - std::exponential_distribution, 2599
 - std::extreme_value_distribution, 2603, 2604
 - std::fisher_f_distribution, 2607, 2608
 - std::gamma_distribution, 2662, 2663
 - std::geometric_distribution, 2667
 - std::lognormal_distribution, 2872, 2873
 - std::negative_binomial_distribution, 3015
 - std::normal_distribution, 3021, 3022
 - std::piecewise_constant_distribution, 3132
 - std::piecewise_linear_distribution, 3138
 - std::poisson_distribution, 3150
 - std::student_t_distribution, 3275, 3276
 - std::uniform_int_distribution, 3342
 - std::uniform_real_distribution, 3346, 3347
 - std::weibull_distribution, 3396, 3397
- partial_sort
 - sorting_algorithms, 225, 226
- partial_sort_copy
 - sorting_algorithms, 226, 227
- partial_sort_minimal_n
 - __gnu_parallel::_Settings, 1266
- partial_sum
 - std, 674
- partial_sum.h, 3776
- partial_sum_dilation
 - __gnu_parallel::_Settings, 1266
- partial_sum_minimal_n
 - __gnu_parallel::_Settings, 1266
- partition
 - mutating_algorithms, 181
- partition.h, 3777
 - _GLIBCXX_VOLATILE, 3777
- partition_chunk_share
 - __gnu_parallel::_Settings, 1266
- partition_chunk_size
 - __gnu_parallel::_Settings, 1266
- partition_copy
 - mutating_algorithms, 181
- partition_minimal_n
 - __gnu_parallel::_Settings, 1267
- partition_point
 - mutating_algorithms, 182
- pbackfail
 - __gnu_cxx::enc_filebuf, 897
 - __gnu_cxx::stdio_filebuf, 978
 - __gnu_cxx::stdio_sync_filebuf, 1007
 - std::basic_filebuf, 1655
 - std::basic_streambuf, 2178
 - std::basic_stringbuf, 2261
- pbase
 - __gnu_cxx::enc_filebuf, 897
 - __gnu_cxx::stdio_filebuf, 979
 - __gnu_cxx::stdio_sync_filebuf, 1007
 - std::basic_filebuf, 1656
 - std::basic_streambuf, 2178
 - std::basic_stringbuf, 2262
- pbump
 - __gnu_cxx::enc_filebuf, 898

- __gnu_cxx::stdio_filebuf, 979
- __gnu_cxx::stdio_sync_filebuf, 1008
- std::basic_filebuf, 1656
- std::basic_streambuf, 2179
- std::basic_stringbuf, 2262
- peek
 - std::basic_fstream, 1723
 - std::basic_ifstream, 1782
 - std::basic_iostream, 1880
 - std::basic_istream, 1937
 - std::basic_istreamstream, 1995
 - std::basic_stringstream, 2325
- pod_char_traits.h, 3778
- pointer
 - std::back_insert_iterator, 1634
 - std::front_insert_iterator, 2638
 - std::insert_iterator, 2730
 - std::istream_iterator, 2800
 - std::istreambuf_iterator, 2804
 - std::iterator, 2808
 - std::ostream_iterator, 3111
 - std::ostreambuf_iterator, 3116
 - std::raw_storage_iterator, 3182
 - std::reverse_iterator, 3216
 - std::set, 3230
- Pointer Abstractions, 65
- pointer.h, 3779
- pointer_abstractions
 - allocate_shared, 67
 - get_deleter, 67
 - make_shared, 67
 - operator<<, 68
- pointer_adaptors
 - ptr_fun, 270
- Poisson Distributions, 300
- polar
 - complex_numbers, 51
- Policy-Based Data Structures, 286
- pool_allocator.h, 3782
- pop
 - std::priority_queue, 3155
 - std::queue, 3163
 - std::stack, 3271
- pop_back
 - __gnu_parallel::-
 - RestrictedBoundedConcurrentQueue, 1257
 - std::deque, 2566
 - std::list, 2842
 - std::vector, 3385
- pop_front
 - __gnu_parallel::-
 - RestrictedBoundedConcurrentQueue, 1257
 - std::deque, 2566
 - std::forward_list, 2628
 - std::list, 2842
- pop_heap
 - heap_algorithms, 276, 277
- pos_format
 - std::moneypunct, 2954
 - std::moneypunct_byname, 2965
- pos_type
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 969
 - __gnu_cxx::stdio_sync_filebuf, 1002
 - std::basic_filebuf, 1647
 - std::basic_fstream, 1686
 - std::basic_ifstream, 1755
 - std::basic_ios, 1810
 - std::basic_iostream, 1845
 - std::basic_istream, 1912
 - std::basic_istreamstream, 1969
 - std::basic_ofstream, 2025
 - std::basic_ostream, 2071
 - std::basic_ostringstream, 2119
 - std::basic_streambuf, 2173
 - std::basic_stringbuf, 2256
 - std::basic_stringstream, 2289
- position
 - std::match_results, 2907
- positive_sign
 - std::moneypunct, 2954
 - std::moneypunct_byname, 2966
- postypes.h, 3783
- pow
 - complex_numbers, 51
- power
 - SGIextensions, 22

- pptr
 - __gnu_cxx::enc_filebuf, 898
 - __gnu_cxx::stdio_filebuf, 980
 - __gnu_cxx::stdio_sync_filebuf, 1008
 - std::basic_filebuf, 1657
 - std::basic_streambuf, 2179
 - std::basic_stringbuf, 2263
- precision
 - std::basic_fstream, 1724
 - std::basic_ifstream, 1783
 - std::basic_ios, 1819
 - std::basic_iostream, 1880
 - std::basic_istream, 1937, 1938
 - std::basic_istream, 1995, 1996
 - std::basic_ofstream, 2046
 - std::basic_ostream, 2091
 - std::basic_ostringstream, 2139
 - std::basic_stringstream, 2325
 - std::ios_base, 2743, 2744
- prefix
 - std::match_results, 2908
- prev_permutation
 - sorting_algorithms, 227, 228
- priority_queue
 - std::priority_queue, 3154
- priority_queue.hpp, 3784
- priority_queue_base_dispatch.hpp, 3785
- probabilities
 - std::discrete_distribution, 2580
- profiler.h, 3786
- profiler_hashtable_size.h, 3790
- profiler_list_to_slist.h, 3791
- profiler_list_to_vector.h, 3792
- profiler_map_to_unordered_map.h, 3793
- profiler_node.h, 3795
- profiler_state.h, 3797
- profiler_trace.h, 3798
- profiler_vector_size.h, 3801
- profiler_vector_to_list.h, 3802
- ptr_fun
 - pointer_adaptors, 270
- pubimbue
 - __gnu_cxx::enc_filebuf, 898
 - __gnu_cxx::stdio_filebuf, 980
 - __gnu_cxx::stdio_sync_filebuf, 1009
- std::basic_filebuf, 1657
- std::basic_streambuf, 2180
- std::basic_stringbuf, 2263
- pubseekoff
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1009
 - std::basic_filebuf, 1658
 - std::basic_streambuf, 2180
 - std::basic_stringbuf, 2264
- pubseekpos
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1009
 - std::basic_filebuf, 1658
 - std::basic_streambuf, 2180
 - std::basic_stringbuf, 2264
- pubsetbuf
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1010
 - std::basic_filebuf, 1658
 - std::basic_streambuf, 2181
 - std::basic_stringbuf, 2264
- pubsync
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1010
 - std::basic_filebuf, 1659
 - std::basic_streambuf, 2181
 - std::basic_stringbuf, 2265
- push
 - std::priority_queue, 3156
 - std::queue, 3164
 - std::stack, 3272
- push_back
 - __gnu_cxx::__versa_string, 842
 - __gnu_debug::basic_string, 1111
 - std::basic_string, 2238
 - std::deque, 2567

- std::list, 2842
- std::vector, 3385
- push_front
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1257
 - std::deque, 2567
 - std::forward_list, 2628
 - std::list, 2842
- push_heap
 - heap_algorithms, 277
- put
 - std::basic_fstream, 1724
 - std::basic_ostream, 1881
 - std::basic_ofstream, 2047
 - std::basic_ostringstream, 2091
 - std::basic_stringstream, 2140
 - std::basic_stringstream, 2326
 - std::money_put, 2942
 - std::num_put, 3044–3047
 - std::time_put, 3315
 - std::time_put_byname, 3319, 3320
- put_money
 - std, 674
- putback
 - std::basic_fstream, 1725
 - std::basic_ifstream, 1783
 - std::basic_ostream, 1881
 - std::basic_istream, 1938
 - std::basic_istream, 1996
 - std::basic_stringstream, 2326
- pword
 - std::basic_fstream, 1726
 - std::basic_ifstream, 1784
 - std::basic_ios, 1819
 - std::basic_ostream, 1882
 - std::basic_istream, 1939
 - std::basic_istream, 1997
 - std::basic_ofstream, 2047
 - std::basic_ostringstream, 2092
 - std::basic_ostringstream, 2140
 - std::basic_stringstream, 2327
 - std::ios_base, 2744
- qsb_steals
 - __gnu_parallel::_Settings, 1267
- queue, 3804
 - std::queue, 3162
- queue.h, 3805
 - _GLIBCXX_VOLATILE, 3805
- quicksort.h, 3806
- quiet_NaN
 - std::numeric_limits, 3051
- radix
 - std::__numeric_limits_base, 1489
 - std::numeric_limits, 3055
- random, 3807
 - generate_canonical, 261
- Random Number Distributions, 291
- Random Number Generation, 261
- Random Number Generators, 288
- Random Number Utilities, 305
- random.h, 3808
- random.tcc, 3813
- random_distributions_bernoulli
 - operator<<, 298
 - operator>>, 298, 299
- random_distributions_normal
 - operator<<, 296
 - operator>>, 296
- random_distributions_poisson
 - operator<<, 301, 302
 - operator>>, 302, 303
- random_distributions_uniform
 - operator<<, 292, 293
 - operator>>, 293, 294
- random_generators
 - minstd_rand, 289
 - minstd_rand0, 289
 - mt19937, 289
 - mt19937_64, 290
 - operator<<, 290
- random_number.h, 3818
- random_sample
 - SGIextensions, 22, 23
- random_sample_n
 - SGIextensions, 23
- random_shuffle
 - mutating_algorithms, 182, 183
- random_shuffle.h, 3819
- random_shuffle_minimal_n

- __gnu_parallel::_Settings, 1267
- ratio, 3821
- Rational Arithmetic, 73
- rb_tree, 3823
- rbegin
 - __gnu_cxx::__versa_string, 842
 - __gnu_debug::basic_string, 1111, 1112
 - std::basic_string, 2238
 - std::deque, 2567, 2568
 - std::list, 2843
 - std::map, 2894, 2895
 - std::multimap, 2985, 2986
 - std::multiset, 3005
 - std::set, 3242
 - std::vector, 3386
- rc_string_base.h, 3824
- rdbuf
 - std::basic_fstream, 1726, 1727
 - std::basic_ifstream, 1784, 1785
 - std::basic_ios, 1820
 - std::basic_iostream, 1882, 1883
 - std::basic_istream, 1939, 1940
 - std::basic_istreamstream, 1997, 1998
 - std::basic_ofstream, 2048
 - std::basic_ostream, 2092, 2093
 - std::basic_ostringstream, 2141
 - std::basic_stringstream, 2327, 2328
- rdstate
 - std::basic_fstream, 1727
 - std::basic_ifstream, 1785
 - std::basic_ios, 1821
 - std::basic_iostream, 1884
 - std::basic_istream, 1940
 - std::basic_istreamstream, 1998
 - std::basic_ofstream, 2048
 - std::basic_ostream, 2094
 - std::basic_ostringstream, 2141
 - std::basic_stringstream, 2328
- read
 - std::basic_fstream, 1727
 - std::basic_ifstream, 1785
 - std::basic_iostream, 1884
 - std::basic_istream, 1941
 - std::basic_istreamstream, 1998
 - std::basic_stringstream, 2329
- readsome
 - std::basic_fstream, 1728
 - std::basic_ifstream, 1786
 - std::basic_iostream, 1885
 - std::basic_istream, 1941
 - std::basic_istreamstream, 1999
 - std::basic_stringstream, 2329
- ref
 - std, 675
- reference
 - std::back_insert_iterator, 1634
 - std::front_insert_iterator, 2638
 - std::insert_iterator, 2730
 - std::istream_iterator, 2800
 - std::istreambuf_iterator, 2804
 - std::iterator, 2808
 - std::ostream_iterator, 3111
 - std::ostreambuf_iterator, 3116
 - std::raw_storage_iterator, 3182
 - std::reverse_iterator, 3216
 - std::set, 3230
- regex, 3825, 3826
 - tr1_regex, 130
- regex_error
 - std::regex_error, 3189
- regex_iterator
 - std::regex_iterator, 3191
- regex_match
 - tr1_regex, 149–153
- regex_replace
 - tr1_regex, 153, 154
- regex_search
 - tr1_regex, 155–158
- regex_token_iterator
 - std::regex_token_iterator, 3196–3198
- regex_traits
 - std::regex_traits, 3202
- register_callback
 - std::basic_fstream, 1729
 - std::basic_ifstream, 1787
 - std::basic_ios, 1821
 - std::basic_iostream, 1885
 - std::basic_istream, 1942
 - std::basic_istreamstream, 2000
 - std::basic_ofstream, 2049

- std::basic_ostream, 2094
- std::basic_ostringstream, 2142
- std::basic_stringstream, 2330
- std::ios_base, 2744
- Regular Expressions, 124
- release
 - std::auto_ptr, 1630
- remove
 - mutating_algorithms, 183
 - std::forward_list, 2629
 - std::list, 2843
- remove_copy
 - mutating_algorithms, 184
- remove_copy_if
 - mutating_algorithms, 184
- remove_if
 - mutating_algorithms, 185
 - std::forward_list, 2629
 - std::list, 2844
- rend
 - __gnu_cxx::__versa_string, 842, 843
 - __gnu_debug::basic_string, 1112
 - std::basic_string, 2239
 - std::deque, 2568
 - std::list, 2844
 - std::map, 2895
 - std::multimap, 2986
 - std::multiset, 3006
 - std::set, 3243
 - std::vector, 3386
- replace
 - __gnu_cxx::__versa_string, 843–849
 - __gnu_debug::basic_string, 1113–1119
 - mutating_algorithms, 185
 - std::basic_string, 2239–2245
- replace_copy
 - std, 675
- replace_copy_if
 - mutating_algorithms, 186
- replace_if
 - mutating_algorithms, 186
- replace_minimal_n
 - __gnu_parallel::_Settings, 1267
- requested_size
 - __gnu_cxx::temporary_buffer, 1026
 - std::_Temporary_buffer, 1578
- reserve
 - __gnu_cxx::__versa_string, 850
 - __gnu_debug::basic_string, 1119
 - std::basic_string, 2246
 - std::vector, 3386
- reset
 - std::auto_ptr, 1630
 - std::bernoulli_distribution, 2350
 - std::binomial_distribution, 2363
 - std::bitset, 2377, 2378
 - std::cauchy_distribution, 2382
 - std::chi_squared_distribution, 2392
 - std::discrete_distribution, 2580
 - std::exponential_distribution, 2599
 - std::extreme_value_distribution, 2604
 - std::fisher_f_distribution, 2608
 - std::gamma_distribution, 2663
 - std::geometric_distribution, 2668
 - std::lognormal_distribution, 2873
 - std::negative_binomial_distribution, 3015
 - std::normal_distribution, 3022
 - std::piecewise_constant_distribution, 3132
 - std::piecewise_linear_distribution, 3138
 - std::poisson_distribution, 3150
 - std::student_t_distribution, 3276
 - std::uniform_int_distribution, 3343
 - std::uniform_real_distribution, 3347
 - std::weibull_distribution, 3397
- resetiosflags
 - std, 676
- resize
 - __gnu_cxx::__versa_string, 851
 - __gnu_debug::basic_string, 1120
 - numeric_arrays, 113
 - std::basic_string, 2247
 - std::deque, 2568
 - std::forward_list, 2629, 2630
 - std::list, 2844
 - std::vector, 3387

-
- result_type
 - __gnu_cxx::__detail::_Ffit_finder, 776
 - __gnu_cxx::binary_compose, 875
 - __gnu_cxx::project1st, 938
 - __gnu_cxx::project2nd, 940
 - __gnu_cxx::select1st, 959
 - __gnu_cxx::select2nd, 960
 - __gnu_cxx::subtractive_rng, 1022
 - __gnu_cxx::unary_compose, 1039
 - __gnu_parallel::_EqualFromLess, 1186
 - __gnu_parallel::_EqualTo, 1188
 - __gnu_parallel::_Less, 1200
 - __gnu_parallel::_Lexicographic, 1202
 - __gnu_parallel::_-
 - LexicographicReverse, 1204
 - _Multiplies, 1240
 - _Plus, 1244
 - _binder1st, 1133
 - _binder2nd, 1135
 - _unary_negate, 1178
 - std::_Maybe_unary_or_binary_-
 - function< _Res, _T1 >, 1556
 - std::_Maybe_unary_or_binary_-
 - function< _Res, _T1, _T2 >, 1558
 - std::bernoulli_distribution, 2348
 - std::binary_function, 2354
 - std::binary_negate, 2355
 - std::binder1st, 2358
 - std::binder2nd, 2360
 - std::binomial_distribution, 2362
 - std::cauchy_distribution, 2382
 - std::chi_squared_distribution, 2391
 - std::const_mem_fun1_ref_t, 2452
 - std::const_mem_fun1_t, 2454
 - std::const_mem_fun_ref_t, 2456
 - std::const_mem_fun_t, 2458
 - std::discard_block_engine, 2572
 - std::discrete_distribution, 2579
 - std::divides, 2584
 - std::equal_to, 2590
 - std::exponential_distribution, 2598
 - std::extreme_value_distribution, 2603
 - std::fisher_f_distribution, 2607
 - std::gamma_distribution, 2661
 - std::geometric_distribution, 2667
 - std::greater, 2671
 - std::greater_equal, 2673
 - std::hash, 2686
 - std::hash< ::bitset< _Nb > >, 2689
 - std::hash< ::vector< bool, _Alloc > >, 2691
 - std::hash< __debug::bitset< _Nb > >, 2693
 - std::hash< __debug::vector< bool, _Alloc > >, 2695
 - std::hash< __gnu_cxx::throw_-value_limit >, 2697
 - std::hash< __gnu_cxx::throw_-value_random >, 2699
 - std::hash< __profile::bitset< _Nb > >, 2701
 - std::hash< __profile::vector< bool, _Alloc > >, 2703
 - std::hash< _Tp * >, 2705
 - std::hash< error_code >, 2707
 - std::hash< string >, 2709
 - std::hash< thread::id >, 2711
 - std::hash< u16string >, 2713
 - std::hash< u32string >, 2715
 - std::hash< wstring >, 2717
 - std::independent_bits_engine, 2720
 - std::less, 2815
 - std::less_equal, 2817
 - std::linear_congruential_engine, 2819
 - std::logical_and, 2866
 - std::logical_not, 2868
 - std::logical_or, 2870
 - std::lognormal_distribution, 2872
 - std::mem_fun1_ref_t, 2911
 - std::mem_fun1_t, 2913
 - std::mem_fun_ref_t, 2915
 - std::mem_fun_t, 2917
 - std::minus, 2927
 - std::modulus, 2929
 - std::multiplies, 2990
-

- std::negate, 3012
- std::negative_binomial_distribution, 3014
- std::normal_distribution, 3020
- std::not_equal_to, 3026
- std::piecewise_constant_distribution, 3131
- std::piecewise_linear_distribution, 3137
- std::plus, 3143
- std::pointer_to_binary_function, 3145
- std::pointer_to_unary_function, 3147
- std::poisson_distribution, 3149
- std::random_device, 3166
- std::seed_seq, 3224
- std::shuffle_order_engine, 3261
- std::student_t_distribution, 3275
- std::unary_function, 3335
- std::unary_negate, 3337
- std::uniform_int_distribution, 3341
- std::uniform_real_distribution, 3346
- std::weibull_distribution, 3396
- rethrow_exception
 - exceptions, 35
- rethrow_if_nested
 - exceptions, 35
- return_temporary_buffer
 - std, 676
- reverse
 - mutating_algorithms, 187
 - std::forward_list, 2630
 - std::list, 2845
- reverse_copy
 - mutating_algorithms, 187
- reverse_iterator
 - std::reverse_iterator, 3216, 3217
 - std::set, 3230
- rfind
 - __gnu_cxx::__versa_string, 852, 853
 - __gnu_debug::basic_string, 1120–1122
 - std::basic_string, 2248, 2249
- riemann_zeta
- tr1_math_spec_func, 122
- right
 - std, 676
 - std::basic_fstream, 1743
 - std::basic_ifstream, 1800
 - std::basic_ios, 1831
 - std::basic_iostream, 1900
 - std::basic_istream, 1955
 - std::basic_istreamstream, 2013
 - std::basic_ofstream, 2060
 - std::basic_ostream, 2106
 - std::basic_ostreamstream, 2154
 - std::basic_stringstream, 2345
 - std::ios_base, 2752
- rope, 3835
- ropeimpl.h, 3840
- rotate
 - mutating_algorithms, 188
- rotate_copy
 - mutating_algorithms, 188
- round_indeterminate
 - std, 600
- round_to_nearest
 - std, 600
- round_toward_infinity
 - std, 600
- round_toward_neg_infinity
 - std, 600
- round_toward_zero
 - std, 600
- round_error
 - std::numeric_limits, 3051
- round_style
 - std::__numeric_limits_base, 1489
 - std::numeric_limits, 3055
- runtime_error
 - std::runtime_error, 3222
- safe_base.h, 3841
- safe_iterator.h, 3842
- safe_iterator.tcc, 3844
- safe_sequence.h, 3845
- sbumpc
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 982

- __gnu_cxx::stdio_sync_filebuf, 1010
- std::basic_filebuf, 1659
- std::basic_streambuf, 2181
- std::basic_stringbuf, 2265
- scan_is
 - std::__ctype_abstract_base, 1402
 - std::ctype, 2469
 - std::ctype< char >, 2483
 - std::ctype< wchar_t >, 2499
 - std::ctype_byname, 2514
 - std::ctype_byname< char >, 2527
- scan_not
 - std::__ctype_abstract_base, 1403
 - std::ctype, 2470
 - std::ctype< char >, 2483
 - std::ctype< wchar_t >, 2499
 - std::ctype_byname, 2515
 - std::ctype_byname< char >, 2527
- scientific
 - std, 677
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1831
 - std::basic_iostream, 1900
 - std::basic_istream, 1955
 - std::basic_istream, 1955
 - std::basic_istringstream, 2013
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostream, 2154
 - std::basic_stringstream, 2346
 - std::ios_base, 2752
- search
 - non_mutating_algorithms, 206, 207
- search.h, 3846
- search_minimal_n
 - __gnu_parallel::_Settings, 1267
- search_n
 - non_mutating_algorithms, 207, 208
- second
 - __gnu_parallel::_IteratorPair, 1193
 - std::pair, 3129
 - std::sub_match, 3283
- second_argument_type
 - __gnu_cxx::project1st, 938
 - __gnu_cxx::project2nd, 940
- __gnu_parallel::_EqualFromLess, 1186
- __gnu_parallel::_EqualTo, 1188
- __gnu_parallel::_Less, 1200
- __gnu_parallel::_Lexicographic, 1202
- __gnu_parallel::_-
 - LexicographicReverse, 1204
- __gnu_parallel::_Multiplies, 1240
- __gnu_parallel::_Plus, 1244
- std::_Maybe_unary_or_binary_-
 - function< _Res, _T1, _T2 >, 1558
- std::binary_function, 2354
- std::binary_negate, 2356
- std::const_mem_fun1_ref_t, 2452
- std::const_mem_fun1_t, 2454
- std::divides, 2584
- std::equal_to, 2590
- std::greater, 2671
- std::greater_equal, 2673
- std::less, 2815
- std::less_equal, 2817
- std::logical_and, 2866
- std::logical_or, 2870
- std::mem_fun1_ref_t, 2911
- std::mem_fun1_t, 2913
- std::minus, 2927
- std::modulus, 2929
- std::multiplies, 2990
- std::not_equal_to, 3026
- std::plus, 3143
- std::pointer_to_binary_function, 3145
- second_type
 - __gnu_parallel::_IteratorPair, 1193
 - std::pair, 3128
 - std::sub_match, 3280
- seconds
 - std::chrono, 730
- seed
 - std::discard_block_engine, 2575, 2576
 - std::independent_bits_engine, 2723
 - std::linear_congruential_engine, 2821

- std::shuffle_order_engine, 3264
- seed_seq
 - std::seed_seq, 3224
- seekdir
 - std::basic_fstream, 1687
 - std::basic_ifstream, 1756
 - std::basic_ios, 1810
 - std::basic_istream, 1845
 - std::basic_istream, 1913
 - std::basic_istream, 1970
 - std::basic_ofstream, 2025
 - std::basic_ostream, 2072
 - std::basic_ostringstream, 2119
 - std::basic_stringstream, 2290
 - std::ios_base, 2741
- seekg
 - std::basic_fstream, 1729, 1730
 - std::basic_ifstream, 1787, 1788
 - std::basic_istream, 1886
 - std::basic_istream, 1942, 1943
 - std::basic_istream, 2000, 2001
 - std::basic_stringstream, 2330, 2331
- seekoff
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 982
 - __gnu_cxx::stdio_sync_filebuf, 1011
 - std::basic_filebuf, 1659
 - std::basic_streambuf, 2182
 - std::basic_stringbuf, 2265
- seekp
 - std::basic_fstream, 1730, 1731
 - std::basic_istream, 1887
 - std::basic_ofstream, 2049, 2050
 - std::basic_ostream, 2094, 2095
 - std::basic_ostringstream, 2142, 2143
 - std::basic_stringstream, 2332
- seekpos
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 983
 - __gnu_cxx::stdio_sync_filebuf, 1011
 - std::basic_filebuf, 1660
 - std::basic_streambuf, 2182
 - std::basic_stringbuf, 2266
- sentry
 - std::basic_istream::sentry, 1958
 - std::basic_ostream::sentry, 2108
- Sequences, 26
- sequential
 - __gnu_parallel, 401
- set, 3847–3849
 - __gnu_parallel::_Settings, 1262
 - std::bitset, 2378
 - std::set, 3231–3233
- Set Operation Algorithms, 232
- set.h, 3850, 3852
- set_algorithms
 - includes, 233
 - set_difference, 234, 235
 - set_intersection, 235, 236
 - set_symmetric_difference, 236, 237
 - set_union, 238
- set_difference
 - set_algorithms, 234, 235
- set_difference_minimal_n
 - __gnu_parallel::_Settings, 1267
- set_intersection
 - set_algorithms, 235, 236
- set_intersection_minimal_n
 - __gnu_parallel::_Settings, 1267
- set_new_handler
 - std, 677
- set_num_threads
 - __gnu_parallel::balanced_
quicksort_tag, 1274
 - __gnu_parallel::balanced_tag, 1276
 - __gnu_parallel::default_parallel_
tag, 1279
 - __gnu_parallel::exact_tag, 1282
 - __gnu_parallel::multiway_
mergesort_exact_tag, 1286
 - __gnu_parallel::multiway_
mergesort_sampling_tag,
1288
 - __gnu_parallel::multiway_
mergesort_tag, 1290
 - __gnu_parallel::omp_loop_static_
tag, 1292
 - __gnu_parallel::omp_loop_tag,
1294
 - __gnu_parallel::parallel_tag, 1297

- __gnu_parallel::quicksort_tag, 1299
 - __gnu_parallel::sampling_tag, 1301
 - __gnu_parallel::unbalanced_tag, 1304
- set_operations.h, 3854
- set_symmetric_difference
 - set_algorithms, 236, 237
- set_symmetric_difference_minimal_n
 - __gnu_parallel::_Settings, 1268
- set_terminate
 - exceptions, 36
- set_unexpected
 - exceptions, 36
- set_union
 - set_algorithms, 238
- set_union_minimal_n
 - __gnu_parallel::_Settings, 1268
- setbase
 - std, 677
- setbuf
 - __gnu_cxx::enc_filebuf, 901
 - __gnu_cxx::stdio_filebuf, 983
 - __gnu_cxx::stdio_sync_filebuf, 1011
 - std::basic_filebuf, 1660
 - std::basic_streambuf, 2182
 - std::basic_stringbuf, 2266
- setf
 - std::basic_fstream, 1731
 - std::basic_ifstream, 1788, 1789
 - std::basic_ios, 1822
 - std::basic_iostream, 1888
 - std::basic_istream, 1944
 - std::basic_istream, 2001, 2002
 - std::basic_ofstream, 2050
 - std::basic_ostream, 2095, 2096
 - std::basic_ostringstream, 2143
 - std::basic_stringstream, 2333
 - std::ios_base, 2745
- setfill
 - std, 677
- setg
 - __gnu_cxx::enc_filebuf, 901
 - __gnu_cxx::stdio_filebuf, 984
 - __gnu_cxx::stdio_sync_filebuf, 1012
- std::basic_filebuf, 1661
- std::basic_streambuf, 2183
- std::basic_stringbuf, 2267
- setiosflags
 - std, 677
- setp
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 984
 - __gnu_cxx::stdio_sync_filebuf, 1012
 - std::basic_filebuf, 1661
 - std::basic_streambuf, 2183
 - std::basic_stringbuf, 2267
- setprecision
 - std, 678
- setstate
 - std::basic_fstream, 1732
 - std::basic_ifstream, 1789
 - std::basic_ios, 1823
 - std::basic_iostream, 1889
 - std::basic_istream, 1944
 - std::basic_istream, 2002
 - std::basic_ofstream, 2051
 - std::basic_ostream, 2096
 - std::basic_ostringstream, 2144
 - std::basic_stringstream, 2333
- settings.h, 3856
 - _GLIBCXX_PARALLEL_-CONDITION, 3857
- setw
 - std, 678
- sgetc
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 984
 - __gnu_cxx::stdio_sync_filebuf, 1013
 - std::basic_filebuf, 1662
 - std::basic_streambuf, 2184
 - std::basic_stringbuf, 2268
- sgetn
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 985
 - __gnu_cxx::stdio_sync_filebuf, 1013
 - std::basic_filebuf, 1662
 - std::basic_streambuf, 2184

- std::basic_stringbuf, 2268
 - SGL STL extensions, 12
 - SGL extensions
 - _Find_first, 17
 - _Find_next, 17
 - _Unchecked_set, 17
 - __median, 16
 - compose1, 18
 - compose2, 18
 - constant0, 18
 - constant1, 18
 - constant2, 18
 - copy_n, 19
 - distance, 19
 - identity_element, 19, 20
 - iota, 20
 - is_heap, 20
 - is_sorted, 21
 - lexicographical_compare_3way, 21
 - power, 22
 - random_sample, 22, 23
 - random_sample_n, 23
 - uninitialized_copy_n, 24
 - shared_future
 - std::shared_future, 3246
 - std::shared_future< _Res & >, 3249
 - std::shared_future< void >, 3252
 - shared_ptr
 - std::shared_ptr, 3255–3258
 - shared_ptr.h, 3858
 - shared_ptr_base.h, 3860
 - shift
 - numeric_arrays, 113
 - showbase
 - std, 678
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1831
 - std::basic_iostream, 1900
 - std::basic_istream, 1955
 - std::basic_istreamstream, 2013
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostringstream, 2154
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
 - showmanyc
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 985
 - __gnu_cxx::stdio_sync_filebuf, 1013
 - std::basic_filebuf, 1662
 - std::basic_streambuf, 2184
 - std::basic_stringbuf, 2268
 - showpoint
 - std, 678
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1832
 - std::basic_iostream, 1901
 - std::basic_istream, 1955
 - std::basic_istreamstream, 2014
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
 - showpos
 - std, 679
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1832
 - std::basic_iostream, 1901
 - std::basic_istream, 1955
 - std::basic_istreamstream, 2014
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
 - shrink_to_fit
 - __gnu_cxx::__versa_string, 854
 - __gnu_debug::basic_string, 1122
 - std::basic_string, 2249
 - std::deque, 2569
 - std::vector, 3387
 - shuffle_order_engine
 - std::shuffle_order_engine, 3261, 3262
 - signaling_NaN
 - std::numeric_limits, 3051
 - sin
-

- complex_numbers, 51
- sinh
 - complex_numbers, 52
- size
 - __gnu_cxx::__versa_string, 854
 - __gnu_cxx::temporary_buffer, 1026
 - __gnu_debug::basic_string, 1122
 - numeric_arrays, 113, 114
 - std::_Temporary_buffer, 1579
 - std::basic_string, 2250
 - std::bitset, 2378
 - std::deque, 2569
 - std::list, 2845
 - std::map, 2895
 - std::match_results, 2908
 - std::multimap, 2986
 - std::multiset, 3006
 - std::priority_queue, 3156
 - std::queue, 3164
 - std::set, 3243
 - std::stack, 3272
 - std::vector, 3388
- size_type
 - std::set, 3230
- skipws
 - std, 679
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1832
 - std::basic_iostream, 1901
 - std::basic_istream, 1955
 - std::basic_istream, 1955
 - std::basic_istringstream, 2014
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
- sleep_for
 - std::this_thread, 756
- sleep_until
 - std::this_thread, 756
- slice
 - numeric_arrays, 90
- slice_array
 - numeric_arrays, 90
- slice_array.h, 3861
- slist, 3862
- snextc
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 986
 - __gnu_cxx::stdio_sync_filebuf, 1014
 - std::basic_filebuf, 1663
 - std::basic_streambuf, 2185
 - std::basic_stringbuf, 2269
- sort
 - sorting_algorithms, 228, 229
 - std::forward_list, 2630, 2631
 - std::list, 2845
- sort.h, 3864
- sort_heap
 - heap_algorithms, 278
- sort_minimal_n
 - __gnu_parallel::_Settings, 1268
- sort_mwms_oversampling
 - __gnu_parallel::_Settings, 1268
- sort_qs_num_samples_preset
 - __gnu_parallel::_Settings, 1268
- sort_qsb_base_case_maximal_n
 - __gnu_parallel::_Settings, 1268
- Sorting Algorithms, 210
- sorting_algorithms
 - inplace_merge, 213
 - is_sorted, 214
 - is_sorted_until, 215
 - lexicographical_compare, 215, 216
 - max, 216, 217
 - max_element, 217, 218
 - merge, 218, 219
 - min, 219, 220
 - min_element, 221
 - minmax, 221, 222
 - minmax_element, 222
 - next_permutation, 223
 - nth_element, 224, 225
 - partial_sort, 225, 226
 - partial_sort_copy, 226, 227
 - prev_permutation, 227, 228
 - sort, 228, 229
 - stable_sort, 229, 230
- sph_bessel
 - tr1_math_spec_func, 122

- sph_legendre
 - tr1_math_spec_func, 122
- sph_neumann
 - tr1_math_spec_func, 122
- splice
 - std::list, 2846
- splice_after
 - std::forward_list, 2631, 2632
- sputback
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 986
 - __gnu_cxx::stdio_sync_filebuf, 1014
 - std::basic_filebuf, 1663
 - std::basic_streambuf, 2185
 - std::basic_stringbuf, 2269
- sputc
 - __gnu_cxx::enc_filebuf, 904
 - __gnu_cxx::stdio_filebuf, 987
 - __gnu_cxx::stdio_sync_filebuf, 1015
 - std::basic_filebuf, 1664
 - std::basic_streambuf, 2186
 - std::basic_stringbuf, 2270
- sputn
 - __gnu_cxx::enc_filebuf, 904
 - __gnu_cxx::stdio_filebuf, 987
 - __gnu_cxx::stdio_sync_filebuf, 1015
 - std::basic_filebuf, 1664
 - std::basic_streambuf, 2186
 - std::basic_stringbuf, 2270
- sqrt
 - complex_numbers, 52
- sregex_token_iterator
 - tr1_regex, 130
- sso_string_base.h, 3866
- sstream, 3867
- sstream.tcc, 3868
- ssub_match
 - tr1_regex, 131
- stable_partition
 - mutating_algorithms, 189
- stable_sort
 - sorting_algorithms, 229, 230
- stack, 3869
- std::stack, 3271
- standard_policies.hpp, 3870
- start
 - numeric_arrays, 114
- state
 - std::fpos, 2635, 2636
- std, 459
 - _Construct, 614
 - _Destroy, 614, 615
 - _final_insertion_sort, 601
 - _find, 601
 - _find_if, 601, 602
 - _find_if_not, 602
 - _gcd, 602
 - _heap_select, 602, 603
 - _inplace_stable_partition, 603
 - _inplace_stable_sort, 603
 - _insertion_sort, 604
 - _introsort_loop, 604
 - _invoke, 605
 - _lg, 605
 - _merge_adaptive, 605
 - _merge_backward, 606
 - _merge_without_buffer, 606, 607
 - _move_median_first, 607
 - _partition, 607, 608
 - _reverse, 608
 - _rotate, 608, 609
 - _rotate_adaptive, 609
 - _search_n, 609, 610
 - _stable_partition_adaptive, 610
 - _unguarded_insertion_sort, 611
 - _unguarded_linear_insert, 611
 - _unguarded_partition, 612
 - _unguarded_partition_pivot, 612
 - _unique_copy, 613, 614
- accumulate, 615
- adjacent_difference, 616
- advance, 617
- bind, 617
- boolalpha, 618
- cerr, 685
- cin, 685
- clog, 685
- cout, 686
- cref, 618

- dec, 618
- denorm_absent, 600
- denorm_indeterminate, 600
- denorm_present, 600
- distance, 618
- endl, 619
- ends, 619
- fixed, 620
- float_denorm_style, 600
- float_round_style, 600
- flush, 620
- forward, 620
- get_money, 620
- get_temporary_buffer, 621
- getline, 621–623
- has_facet, 624
- hex, 624
- inner_product, 624, 625
- internal, 625
- iota, 625
- isalnum, 626
- isalpha, 626
- iscntrl, 626
- isdigit, 626
- isgraph, 626
- islower, 626
- isprint, 627
- ispunct, 627
- isspace, 627
- isupper, 627
- isxdigit, 627
- left, 627
- new_handler, 598
- noboolalpha, 627
- noshowbase, 628
- noshowpoint, 628
- noshowpos, 628
- noskipws, 628
- nounitbuf, 628
- nouppercase, 628
- oct, 629
- operator<, 635–641
- operator<<, 641–648
- operator<=, 648–651
- operator>, 659–661
- operator>>, 665–672
- operator>=, 662–665
- operator[^], 673
- operator+, 633–635
- operator==, 652–658
- operator&, 633
- partial_sum, 674
- put_money, 674
- ref, 675
- replace_copy, 675
- resetiosflags, 676
- return_temporary_buffer, 676
- right, 676
- round_indeterminate, 600
- round_to_nearest, 600
- round_toward_infinity, 600
- round_toward_neg_infinity, 600
- round_toward_zero, 600
- scientific, 677
- set_new_handler, 677
- setbase, 677
- setfill, 677
- setiosflags, 677
- setprecision, 678
- setw, 678
- showbase, 678
- showpoint, 678
- showpos, 679
- skipws, 679
- streamoff, 598
- streampos, 599
- streamsize, 599
- swap, 679–681
- tolower, 682
- toupper, 682
- u16streampos, 599
- u32streampos, 599
- uninitialized_copy, 682
- uninitialized_copy_n, 682
- uninitialized_fill, 683
- uninitialized_fill_n, 683
- unitbuf, 684
- uppercase, 684
- use_facet, 684
- wcerr, 686
- wcin, 686
- wclog, 686

- wcout, 686
- ws, 685
- wstreampos, 599
- std::__basic_future, 1384
- _M_get_result, 1385
- std::__codecvt_abstract_base, 1386
- do_out, 1388
- in, 1388
- out, 1389
- unshift, 1390
- std::__ctype_abstract_base, 1392
- char_type, 1394
- do_is, 1395
- do_narrow, 1395, 1396
- do_scan_is, 1396
- do_scan_not, 1397
- do_tolower, 1397, 1398
- do_toupper, 1398, 1399
- do_widen, 1399, 1400
- is, 1400, 1401
- narrow, 1401, 1402
- scan_is, 1402
- scan_not, 1403
- tolower, 1403, 1404
- toupper, 1404
- widen, 1405
- std::__debug, 687
- std::__debug::bitset, 1407
- _M_const_iterators, 1410
- _M_detach_all, 1409
- _M_detach_singular, 1409
- _M_get_mutex, 1409
- _M_invalidate_all, 1409
- _M_iterators, 1410
- _M_revalidate_singular, 1410
- _M_swap, 1410
- _M_version, 1411
- std::__debug::deque, 1412
- _M_const_iterators, 1416
- _M_detach_all, 1415
- _M_detach_singular, 1415
- _M_get_mutex, 1415
- _M_invalidate_all, 1415
- _M_invalidate_if, 1416
- _M_iterators, 1417
- _M_revalidate_singular, 1416
- _M_swap, 1416
- _M_transfer_iter, 1416
- _M_version, 1417
- std::__debug::list, 1418
- _M_const_iterators, 1423
- _M_detach_all, 1421
- _M_detach_singular, 1421
- _M_get_mutex, 1421
- _M_invalidate_all, 1421
- _M_invalidate_if, 1422
- _M_iterators, 1423
- _M_revalidate_singular, 1422
- _M_swap, 1422
- _M_transfer_iter, 1422
- _M_version, 1423
- std::__debug::map, 1424
- _M_const_iterators, 1428
- _M_detach_all, 1426
- _M_detach_singular, 1426
- _M_get_mutex, 1427
- _M_invalidate_all, 1427
- _M_invalidate_if, 1427
- _M_iterators, 1428
- _M_revalidate_singular, 1427
- _M_swap, 1428
- _M_transfer_iter, 1428
- _M_version, 1428
- std::__debug::multimap, 1430
- _M_const_iterators, 1434
- _M_detach_all, 1432
- _M_detach_singular, 1432
- _M_get_mutex, 1433
- _M_invalidate_all, 1433
- _M_invalidate_if, 1433
- _M_iterators, 1434
- _M_revalidate_singular, 1433
- _M_swap, 1434
- _M_transfer_iter, 1434
- _M_version, 1434
- std::__debug::multiset, 1436
- _M_const_iterators, 1440
- _M_detach_all, 1438
- _M_detach_singular, 1438
- _M_get_mutex, 1439
- _M_invalidate_all, 1439
- _M_invalidate_if, 1439

- [_M_iterators, 1440](#)
- [_M_revalidate_singular, 1439](#)
- [_M_swap, 1440](#)
- [_M_transfer_iter, 1440](#)
- [_M_version, 1440](#)
- [std::__debug::set, 1442](#)
 - [_M_const_iterators, 1446](#)
 - [_M_detach_all, 1444](#)
 - [_M_detach_singular, 1444](#)
 - [_M_get_mutex, 1445](#)
 - [_M_invalidate_all, 1445](#)
 - [_M_invalidate_if, 1445](#)
 - [_M_iterators, 1446](#)
 - [_M_revalidate_singular, 1445](#)
 - [_M_swap, 1446](#)
 - [_M_transfer_iter, 1446](#)
 - [_M_version, 1446](#)
- [std::__debug::unordered_map, 1448](#)
 - [_M_const_iterators, 1452](#)
 - [_M_detach_all, 1450](#)
 - [_M_detach_singular, 1450](#)
 - [_M_get_mutex, 1450](#)
 - [_M_invalidate_all, 1450](#)
 - [_M_invalidate_if, 1451](#)
 - [_M_iterators, 1452](#)
 - [_M_revalidate_singular, 1451](#)
 - [_M_swap, 1451](#)
 - [_M_transfer_iter, 1451](#)
 - [_M_version, 1452](#)
- [std::__debug::unordered_multimap, 1453](#)
 - [_M_const_iterators, 1457](#)
 - [_M_detach_all, 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_iter, 1456](#)
 - [_M_version, 1457](#)
- [std::__debug::unordered_multiset, 1458](#)
 - [_M_const_iterators, 1461](#)
 - [_M_detach_all, 1460](#)
 - [_M_detach_singular, 1460](#)
 - [_M_get_mutex, 1460](#)
 - [_M_invalidate_all, 1460](#)
 - [_M_invalidate_if, 1461](#)
 - [_M_iterators, 1462](#)
 - [_M_revalidate_singular, 1461](#)
 - [_M_swap, 1461](#)
 - [_M_transfer_iter, 1461](#)
 - [_M_version, 1462](#)
- [std::__debug::unordered_set, 1463](#)
 - [_M_const_iterators, 1467](#)
 - [_M_detach_all, 1465](#)
 - [_M_detach_singular, 1465](#)
 - [_M_get_mutex, 1465](#)
 - [_M_invalidate_all, 1465](#)
 - [_M_invalidate_if, 1466](#)
 - [_M_iterators, 1467](#)
 - [_M_revalidate_singular, 1466](#)
 - [_M_swap, 1466](#)
 - [_M_transfer_iter, 1466](#)
 - [_M_version, 1467](#)
- [std::__debug::vector, 1468](#)
 - [_M_const_iterators, 1473](#)
 - [_M_detach_all, 1471](#)
 - [_M_detach_singular, 1471](#)
 - [_M_get_mutex, 1471](#)
 - [_M_invalidate_all, 1471](#)
 - [_M_invalidate_if, 1472](#)
 - [_M_iterators, 1473](#)
 - [_M_revalidate_singular, 1472](#)
 - [_M_swap, 1472](#)
 - [_M_transfer_iter, 1472](#)
 - [_M_version, 1473](#)
- [vector, 1471](#)
- [std::__declval_protector, 1474](#)
- [std::__detail, 693](#)
- [std::__exception_ptr::exception_ptr, 1475](#)
- [std::__future_base, 1476](#)
- [std::__future_base::_Ptr, 1477](#)
- [std::__future_base::_Result, 1478](#)
- [std::__future_base::_Result< _Res & >, 1479](#)
- [std::__future_base::_Result< void >, 1480](#)
- [std::__future_base::_Result_base, 1481](#)
- [std::__future_base::_State, 1482](#)
- [std::__is_location_invariant, 1483](#)
- [std::__is_member_pointer_helper, 1484](#)

-
- std::__numeric_limits_base, 1485
 - digits, 1486
 - digits10, 1486
 - has_denorm, 1486
 - has_denorm_loss, 1486
 - has_infinity, 1486
 - has_quiet_NaN, 1487
 - has_signaling_NaN, 1487
 - is_bounded, 1487
 - is_exact, 1487
 - is_iec559, 1487
 - is_integer, 1487
 - is_modulo, 1488
 - is_signed, 1488
 - is_specialized, 1488
 - max_digits10, 1488
 - max_exponent, 1488
 - max_exponent10, 1488
 - min_exponent, 1488
 - min_exponent10, 1489
 - radix, 1489
 - round_style, 1489
 - tinyness_before, 1489
 - traps, 1489
 - std::__parallel, 694
 - std::__parallel::__CRandNumber, 1490
 - std::__profile, 720
 - std::__profile::bitset, 1491
 - std::__profile::deque, 1493
 - std::__profile::list, 1496
 - std::__profile::map, 1499
 - std::__profile::multimap, 1502
 - std::__profile::multiset, 1505
 - std::__profile::set, 1507
 - std::__profile::unordered_map, 1509
 - std::__profile::unordered_multimap, 1511
 - std::__profile::unordered_multiset, 1513
 - std::__profile::unordered_set, 1515
 - std::_Base_bitset, 1517
 - _M_w, 1518
 - std::_Base_bitset< 0 >, 1519
 - std::_Base_bitset< 1 >, 1521
 - std::_Build_index_tuple, 1523
 - std::_Deque_base, 1524
 - _M_initialize_map, 1525
 - std::_Deque_iterator, 1527
 - _M_set_node, 1528
 - std::_Derives_from_binary_function, 1529
 - std::_Derives_from_unary_function, 1530
 - std::_Function_base, 1531
 - std::_Function_to_function_pointer, 1533
 - std::_Fwd_list_base, 1534
 - std::_Fwd_list_const_iterator, 1536
 - std::_Fwd_list_iterator, 1538
 - std::_Fwd_list_node, 1539
 - std::_Fwd_list_node_base, 1541
 - std::_Has_result_type_helper, 1543
 - std::_Index_tuple, 1544
 - std::_List_base, 1545
 - std::_List_const_iterator, 1547
 - std::_List_iterator, 1549
 - std::_List_node, 1550
 - _M_data, 1551
 - std::_List_node_base, 1552
 - std::_Maybe_get_result_type, 1553
 - std::_Maybe_unary_or_binary_function, 1554
 - std::_Maybe_unary_or_binary_function< _Res, _T1 >, 1555
 - argument_type, 1555
 - result_type, 1556
 - std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >, 1557
 - first_argument_type, 1558
 - result_type, 1558
 - second_argument_type, 1558
 - std::_Maybe_wrap_member_pointer, 1559
 - std::_Maybe_wrap_member_pointer< _Tp _Class::* >, 1560
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >, 1561
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >, 1563
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >, 1564
-

- std::_Mem_fn< _Res(_Class::*)(_ArgTypes...)>, 1566
 std::_Mu< _Arg, false, false >, 1568
 std::_Mu< _Arg, false, true >, 1569
 std::_Mu< _Arg, true, false >, 1570
 std::_Mu< reference_wrapper< _Tp >, false, false >, 1571
 std::_Placeholder, 1572
 std::_Reference_wrapper_base, 1573
 std::_Safe_tuple_element, 1574
 std::_Safe_tuple_element_impl, 1575
 std::_Safe_tuple_element_impl< __i, _Tuple, false >, 1576
 std::_Temporary_buffer, 1577
 _Temporary_buffer, 1578
 begin, 1578
 end, 1578
 requested_size, 1578
 size, 1579
 std::_Tuple_impl< _Idx >, 1580
 std::_Tuple_impl< _Idx, _Head, _Tail...>, 1581
 std::_Vector_base, 1583
 std::_Weak_result_type, 1585
 std::_Weak_result_type_impl, 1586
 std::_Weak_result_type_impl< _Res(_Res*)(_ArgTypes...)>, 1588
 std::_Weak_result_type_impl< _Res(_Res&)(_ArgTypes...)>, 1587
 std::_Weak_result_type_impl< _Res(_Res(_ArgTypes...))>, 1589
 std::_Weak_result_type_impl< _Res(_Res(_Class::*)(_ArgTypes...)) const >, 1590
 std::_Weak_result_type_impl< _Res(_Res(_Class::*)(_ArgTypes...)) const volatile >, 1591
 std::_Weak_result_type_impl< _Res(_Res(_Class::*)(_ArgTypes...)) volatile >, 1592
 std::_Weak_result_type_impl< _Res(_Res(_Class::*)(_ArgTypes...))>, 1593
 std::add_const, 1594
 std::add_cv, 1595
 std::add_lvalue_reference, 1596
 std::add_pointer, 1597
 std::add_rvalue_reference, 1598
 std::add_volatile, 1599
 std::adopt_lock_t, 1600
 std::aligned_storage, 1601
 std::alignment_of, 1602
 std::allocator, 1603
 std::allocator< void >, 1605
 std::array, 1606
 std::atomic, 1608
 std::atomic< _Tp * >, 1609
 std::atomic< bool >, 1610
 std::atomic< char >, 1611
 std::atomic< char16_t >, 1612
 std::atomic< char32_t >, 1613
 std::atomic< int >, 1614
 std::atomic< long >, 1615
 std::atomic< long long >, 1616
 std::atomic< short >, 1617
 std::atomic< signed char >, 1618
 std::atomic< unsigned char >, 1619
 std::atomic< unsigned int >, 1620
 std::atomic< unsigned long >, 1621
 std::atomic< unsigned long long >, 1622
 std::atomic< unsigned short >, 1623
 std::atomic< void * >, 1624
 std::atomic< wchar_t >, 1625
 std::auto_ptr, 1626
 ~auto_ptr, 1628
 auto_ptr, 1627, 1628
 element_type, 1627
 get, 1629
 operator*, 1629
 operator->, 1629
 operator=, 1629, 1630
 release, 1630
 reset, 1630
 std::auto_ptr_ref, 1632
 std::back_insert_iterator, 1633
 back_insert_iterator, 1635
 container_type, 1634
 difference_type, 1634
 iterator_category, 1634
 operator*, 1635
 operator++, 1635
 operator=, 1636

- pointer, 1634
- reference, 1634
- value_type, 1635
- std::bad_alloc, 1637
 - what, 1637
- std::bad_cast, 1638
 - what, 1638
- std::bad_exception, 1639
 - what, 1639
- std::bad_function_call, 1640
 - what, 1640
- std::bad_typeid, 1641
 - what, 1641
- std::bad_weak_ptr, 1642
 - what, 1642
- std::basic_filebuf, 1643
 - ~basic_filebuf, 1648
 - _M_buf, 1669
 - _M_buf_locale, 1669
 - _M_buf_size, 1669
 - _M_create_pback, 1648
 - _M_destroy_pback, 1648
 - _M_ext_buf, 1669
 - _M_ext_buf_size, 1669
 - _M_ext_next, 1670
 - _M_in_beg, 1670
 - _M_in_cur, 1670
 - _M_in_end, 1671
 - _M_mode, 1671
 - _M_out_beg, 1671
 - _M_out_cur, 1671
 - _M_out_end, 1672
 - _M_pback, 1672
 - _M_pback_cur_save, 1672
 - _M_pback_end_save, 1673
 - _M_pback_init, 1673
 - _M_reading, 1673
 - _M_set_buffer, 1649
 - __streambuf_type, 1646
- basic_filebuf, 1648
- char_type, 1647
- close, 1649
- eback, 1650
- egptr, 1650
- epptr, 1650
- gbump, 1651
- getloc, 1651
- gptr, 1652
- imbue, 1652
- in_avail, 1653
- int_type, 1647
- is_open, 1653
- off_type, 1647
- open, 1653, 1654
- overflow, 1654
- pbackfail, 1655
- pbase, 1656
- pbump, 1656
- pos_type, 1647
- pptr, 1657
- pubimbue, 1657
- pubseekoff, 1658
- pubseekpos, 1658
- pubsetbuf, 1658
- pubsync, 1659
- sbumpc, 1659
- seekoff, 1659
- seekpos, 1660
- setbuf, 1660
- setg, 1661
- setp, 1661
- sgetc, 1662
- sgetn, 1662
- showmanyc, 1662
- snextc, 1663
- sputbackc, 1663
- sputc, 1664
- sputn, 1664
- stossc, 1665
- sungetc, 1665
- sync, 1665
- traits_type, 1648
- uflow, 1666
- underflow, 1666
- xsggetn, 1667
- xsputn, 1668
- std::basic_fstream, 1675
 - ~basic_fstream, 1688
 - _M_gcount, 1738
 - _M_getloc, 1689
 - _M_write, 1689
 - __ctype_type, 1683

__num_get_type, 1683
__num_put_type, 1683
adjustfield, 1739
app, 1739
ate, 1739
bad, 1690
badbit, 1739
basefield, 1739
basic_fstream, 1688
beg, 1740
binary, 1740
boolalpha, 1740
char_type, 1684
clear, 1690
close, 1690
copyfmt, 1690
cur, 1740
dec, 1740
end, 1741
eof, 1691
eofbit, 1741
event, 1687
event_callback, 1684
exceptions, 1691, 1692
fail, 1692
failbit, 1741
fill, 1693
fixed, 1741
flags, 1693, 1694
floatfield, 1742
flush, 1694
fmtflags, 1684
gcount, 1695
get, 1695–1698
getline, 1698, 1699
getloc, 1700
good, 1700
goodbit, 1742
hex, 1742
ignore, 1700, 1701
imbue, 1702
in, 1742
init, 1702
int_type, 1685
internal, 1743
iostate, 1685
is_open, 1703
iword, 1703
left, 1743
narrow, 1703
oct, 1743
off_type, 1686
open, 1704
openmode, 1686
operator void *, 1705
operator<<, 1705–1714
operator>>, 1714–1723
out, 1743
peek, 1723
pos_type, 1686
precision, 1724
put, 1724
putback, 1725
pword, 1726
rdbuf, 1726, 1727
rdstate, 1727
read, 1727
readsome, 1728
register_callback, 1729
right, 1743
scientific, 1744
seekdir, 1687
seekg, 1729, 1730
seekp, 1730, 1731
setf, 1731
setstate, 1732
showbase, 1744
showpoint, 1744
showpos, 1744
skipws, 1744
sync, 1732
sync_with_stdio, 1733
tellg, 1733
tellp, 1734
tie, 1734, 1735
traits_type, 1687
trunc, 1744
unget, 1735
unitbuf, 1744
unsetf, 1736
uppercase, 1745
widen, 1736

- width, 1736, 1737
- write, 1737
- xalloc, 1738
- std::basic_ifstream, 1746
 - ~basic_ifstream, 1757
 - _M_gcount, 1794
 - _M_getloc, 1758
 - __ctype_type, 1752
 - __num_get_type, 1752
 - __num_put_type, 1752
- adjustfield, 1795
- app, 1795
- ate, 1795
- bad, 1758
- badbit, 1795
- basefield, 1796
- basic_ifstream, 1757
- beg, 1796
- binary, 1796
- boolalpha, 1796
- char_type, 1753
- clear, 1758
- close, 1759
- copyfmt, 1759
- cur, 1796
- dec, 1797
- end, 1797
- eof, 1759
- eofbit, 1797
- event, 1756
- event_callback, 1753
- exceptions, 1760
- fail, 1761
- failbit, 1797
- fill, 1761, 1762
- fixed, 1798
- flags, 1762
- floatfield, 1798
- fmtflags, 1753
- gcount, 1763
- get, 1763–1766
- getline, 1766, 1767
- getloc, 1768
- good, 1768
- goodbit, 1798
- hex, 1798
- ignore, 1768, 1769
- imbue, 1770
- in, 1798
- init, 1770
- int_type, 1754
- internal, 1799
- iostate, 1754
- is_open, 1771
- iword, 1771
- left, 1799
- narrow, 1771
- oct, 1799
- off_type, 1755
- open, 1772
- openmode, 1755
- operator void *, 1773
- operator>>, 1773–1782
- out, 1799
- peek, 1782
- pos_type, 1755
- precision, 1783
- putback, 1783
- pword, 1784
- rdbuf, 1784, 1785
- rdstate, 1785
- read, 1785
- readsome, 1786
- register_callback, 1787
- right, 1800
- scientific, 1800
- seekdir, 1756
- seekg, 1787, 1788
- setf, 1788, 1789
- setstate, 1789
- showbase, 1800
- showpoint, 1800
- showpos, 1800
- skipws, 1800
- sync, 1790
- sync_with_stdio, 1790
- tellg, 1791
- tie, 1791, 1792
- traits_type, 1756
- trunc, 1800
- unget, 1792
- unitbuf, 1800

- unsetf, 1792
- uppercase, 1801
- widen, 1793
- width, 1793, 1794
- xalloc, 1794
- std::basic_ios, 1802
 - ~basic_ios, 1811
 - _M_getloc, 1811
 - __ctype_type, 1806
 - __num_get_type, 1806
 - __num_put_type, 1806
 - adjustfield, 1826
 - app, 1826
 - ate, 1826
 - bad, 1812
 - badbit, 1827
 - basefield, 1827
 - basic_ios, 1811
 - beg, 1827
 - binary, 1827
 - boolalpha, 1828
 - char_type, 1806
 - clear, 1812
 - copyfmt, 1812
 - cur, 1828
 - dec, 1828
 - end, 1828
 - eof, 1813
 - eofbit, 1828
 - event, 1811
 - event_callback, 1807
 - exceptions, 1813, 1814
 - fail, 1814
 - failbit, 1829
 - fill, 1815
 - fixed, 1829
 - flags, 1815, 1816
 - floatfield, 1829
 - fmtflags, 1807
 - getloc, 1816
 - good, 1816
 - goodbit, 1829
 - hex, 1830
 - imbue, 1817
 - in, 1830
 - init, 1817
 - int_type, 1808
 - internal, 1830
 - iostate, 1808
 - iword, 1817
 - left, 1831
 - narrow, 1818
 - oct, 1831
 - off_type, 1809
 - openmode, 1809
 - operator void *, 1818
 - out, 1831
 - pos_type, 1810
 - precision, 1819
 - pword, 1819
 - rdbuf, 1820
 - rdstate, 1821
 - register_callback, 1821
 - right, 1831
 - scientific, 1831
 - seekdir, 1810
 - setf, 1822
 - setstate, 1823
 - showbase, 1831
 - showpoint, 1832
 - showpos, 1832
 - skipws, 1832
 - sync_with_stdio, 1823
 - tie, 1824
 - traits_type, 1810
 - trunc, 1832
 - unitbuf, 1832
 - unsetf, 1824
 - uppercase, 1832
 - widen, 1825
 - width, 1825
 - xalloc, 1826
- std::basic_ostream, 1834
 - ~basic_ostream, 1846
 - _M_gcount, 1895
 - _M_getloc, 1846
 - _M_write, 1847
 - __ctype_type, 1841
 - __num_get_type, 1841
 - __num_put_type, 1841, 1842
 - adjustfield, 1895
 - app, 1895

ate, 1896
bad, 1847
badbit, 1896
basefield, 1896
basic_ostream, 1846
beg, 1896
binary, 1897
boolalpha, 1897
char_type, 1842
clear, 1847
copyfmt, 1848
cur, 1897
dec, 1897
end, 1897
eof, 1848
eofbit, 1898
event, 1846
event_callback, 1842
exceptions, 1849
fail, 1850
failbit, 1898
fill, 1850
fixed, 1898
flags, 1851
floatfield, 1898
flush, 1851
fmtflags, 1843
gcount, 1852
get, 1852–1855
getline, 1855, 1856
getloc, 1857
good, 1857
goodbit, 1898
hex, 1899
ignore, 1857, 1858
imbue, 1859
in, 1899
init, 1860
int_type, 1843
internal, 1899
iostate, 1844
iword, 1860
left, 1900
narrow, 1860
oct, 1900
off_type, 1844
openmode, 1844
operator void *, 1861
operator<<, 1861–1870
operator>>, 1871–1880
out, 1900
peek, 1880
pos_type, 1845
precision, 1880
put, 1881
putback, 1881
pword, 1882
rdbuf, 1882, 1883
rdstate, 1884
read, 1884
readsome, 1885
register_callback, 1885
right, 1900
scientific, 1900
seekdir, 1845
seekg, 1886
seekp, 1887
setf, 1888
setstate, 1889
showbase, 1900
showpoint, 1901
showpos, 1901
skipws, 1901
sync, 1889
sync_with_stdio, 1890
tellg, 1890
tellp, 1891
tie, 1891, 1892
traits_type, 1845
trunc, 1901
unget, 1892
unitbuf, 1901
unsetf, 1892
uppercase, 1901
widen, 1893
width, 1893, 1894
write, 1894
xalloc, 1895
std::basic_istream, 1903
~basic_istream, 1914
_M_gcount, 1950
_M_getloc, 1914

__ctype_type, 1909
__num_get_type, 1909
__num_put_type, 1909
adjustfield, 1950
app, 1950
ate, 1950
bad, 1914
badbit, 1950
basefield, 1951
basic_istream, 1914
beg, 1951
binary, 1951
boolalpha, 1951
char_type, 1909
clear, 1915
copyfmt, 1915
cur, 1952
dec, 1952
end, 1952
eof, 1915
eofbit, 1952
event, 1913
event_callback, 1910
exceptions, 1916
fail, 1917
failbit, 1952
fill, 1917, 1918
fixed, 1953
flags, 1918
floatfield, 1953
fmtflags, 1910
gcount, 1919
get, 1919–1922
getline, 1922, 1923
getloc, 1924
good, 1924
goodbit, 1953
hex, 1953
ignore, 1924, 1925
imbue, 1926
in, 1954
init, 1926
int_type, 1911
internal, 1954
iostate, 1911
iword, 1927
left, 1954
narrow, 1927
oct, 1954
off_type, 1912
openmode, 1912
operator void *, 1928
operator>>, 1928–1937
out, 1954
peek, 1937
pos_type, 1912
precision, 1937, 1938
putback, 1938
pword, 1939
rdbuf, 1939, 1940
rdstate, 1940
read, 1941
readsome, 1941
register_callback, 1942
right, 1955
scientific, 1955
seekdir, 1913
seekg, 1942, 1943
setf, 1944
setstate, 1944
showbase, 1955
showpoint, 1955
showpos, 1955
skipws, 1955
sync, 1945
sync_with_stdio, 1946
tellg, 1946
tie, 1946, 1947
traits_type, 1913
trunc, 1956
unget, 1947
unitbuf, 1956
unsetf, 1948
uppercase, 1956
widen, 1948
width, 1949
xalloc, 1949
std::basic_istream::sentry, 1957
operator bool, 1958
sentry, 1958
traits_type, 1957
std::basic_istream, 1960

~basic_istream, 1971
_M_gcount, 2008
_M_getloc, 1972
__ctype_type, 1966
__num_get_type, 1966
__num_put_type, 1966
adjustfield, 2008
app, 2008
ate, 2009
bad, 1972
badbit, 2009
basefield, 2009
basic_istream, 1971
beg, 2009
binary, 2010
boolalpha, 2010
char_type, 1967
clear, 1972
copyfmt, 1973
cur, 2010
dec, 2010
end, 2010
eof, 1973
eofbit, 2011
event, 1970
event_callback, 1967
exceptions, 1974
fail, 1975
failbit, 2011
fill, 1975, 1976
fixed, 2011
flags, 1976
floatfield, 2011
fmtflags, 1967
gcount, 1977
get, 1977–1980
getline, 1980, 1981
getloc, 1982
good, 1982
goodbit, 2011
hex, 2012
ignore, 1982, 1983
imbue, 1984
in, 2012
init, 1984
int_type, 1968
internal, 2012
iostate, 1968
iword, 1985
left, 2013
narrow, 1985
oct, 2013
off_type, 1969
openmode, 1969
operator void *, 1986
operator>>, 1986–1995
out, 2013
peek, 1995
pos_type, 1969
precision, 1995, 1996
putback, 1996
pword, 1997
rdbuf, 1997, 1998
rdstate, 1998
read, 1998
readsome, 1999
register_callback, 2000
right, 2013
scientific, 2013
seekdir, 1970
seekg, 2000, 2001
setf, 2001, 2002
setstate, 2002
showbase, 2013
showpoint, 2014
showpos, 2014
skipws, 2014
str, 2003
sync, 2003
sync_with_stdio, 2004
tellg, 2004
tie, 2005
traits_type, 1970
trunc, 2014
unget, 2006
unitbuf, 2014
unsetf, 2006
uppercase, 2014
widen, 2006
width, 2007
xalloc, 2008
std::basic_ofstream, 2016

~basic_ofstream, 2027
_M_getloc, 2027
_M_write, 2027
__ctype_type, 2022
__num_get_type, 2022
__num_put_type, 2022
adjustfield, 2056
app, 2056
ate, 2056
bad, 2028
badbit, 2056
basefield, 2056
basic_ofstream, 2026, 2027
beg, 2057
binary, 2057
boolalpha, 2057
char_type, 2022
clear, 2028
close, 2029
copyfmt, 2029
cur, 2057
dec, 2057
end, 2058
eof, 2029
eofbit, 2058
event, 2026
event_callback, 2022
exceptions, 2030
fail, 2031
failbit, 2058
fill, 2031
fixed, 2058
flags, 2032
floatfield, 2059
flush, 2032
fmtflags, 2023
getloc, 2033
good, 2033
goodbit, 2059
hex, 2059
imbue, 2033
in, 2059
init, 2034
int_type, 2024
internal, 2060
iostate, 2024
is_open, 2034
iword, 2034
left, 2060
narrow, 2035
oct, 2060
off_type, 2024
open, 2035, 2036
openmode, 2024
operator void *, 2036
operator<<, 2037–2046
out, 2060
pos_type, 2025
precision, 2046
put, 2047
pword, 2047
rdbuf, 2048
rdstate, 2048
register_callback, 2049
right, 2060
scientific, 2061
seekdir, 2025
seekp, 2049, 2050
setf, 2050
setstate, 2051
showbase, 2061
showpoint, 2061
showpos, 2061
skipws, 2061
sync_with_stdio, 2051
tellp, 2052
tie, 2052, 2053
traits_type, 2025
trunc, 2061
unitbuf, 2061
unsetf, 2053
uppercase, 2062
widen, 2053
width, 2054
write, 2054
xalloc, 2055
std::basic_ostream, 2063
~basic_ostream, 2073
_M_getloc, 2073
_M_write, 2073
__ctype_type, 2068
__num_get_type, 2068

__num_put_type, 2068
adjustfield, 2101
app, 2101
ate, 2101
bad, 2074
badbit, 2101
basefield, 2102
basic_ostream, 2073
beg, 2102
binary, 2102
boolalpha, 2102
char_type, 2069
clear, 2074
copyfmt, 2075
cur, 2102
dec, 2103
end, 2103
eof, 2075
eofbit, 2103
event, 2072
event_callback, 2069
exceptions, 2075, 2076
fail, 2076
failbit, 2103
fill, 2077
fixed, 2104
flags, 2078
floatfield, 2104
flush, 2078
fmtflags, 2069
getloc, 2079
good, 2079
goodbit, 2104
hex, 2104
imbue, 2079
in, 2104
init, 2080
int_type, 2070
internal, 2105
iostate, 2070
iword, 2080
left, 2105
narrow, 2081
oct, 2105
off_type, 2071
openmode, 2071
operator void *, 2081
operator<<, 2081–2090
out, 2105
pos_type, 2071
precision, 2091
put, 2091
pword, 2092
rdbuf, 2092, 2093
rdstate, 2094
register_callback, 2094
right, 2106
scientific, 2106
seekdir, 2072
seekp, 2094, 2095
setf, 2095, 2096
setstate, 2096
showbase, 2106
showpoint, 2106
showpos, 2106
skipws, 2106
sync_with_stdio, 2097
tellp, 2097
tie, 2097, 2098
traits_type, 2072
trunc, 2106
unitbuf, 2106
unsetf, 2098
uppercase, 2107
widen, 2098
width, 2099
write, 2100
xalloc, 2100
std::basic_ostream::sentry, 2108
 ~sentry, 2108
 operator bool, 2109
 sentry, 2108
std::basic_ostringstream, 2110
 ~basic_ostringstream, 2121
 _M_getloc, 2121
 _M_write, 2122
 __ctype_type, 2116
 __num_get_type, 2116
 __num_put_type, 2116
 adjustfield, 2149
 app, 2149
 ate, 2149

bad, 2122
 badbit, 2150
 basefield, 2150
 basic_ostringstream, 2120
 beg, 2150
 binary, 2150
 boolalpha, 2151
 char_type, 2116
 clear, 2122
 copyfmt, 2123
 cur, 2151
 dec, 2151
 end, 2151
 eof, 2123
 eofbit, 2151
 event, 2120
 event_callback, 2116
 exceptions, 2124
 fail, 2125
 failbit, 2152
 fill, 2125
 fixed, 2152
 flags, 2126
 floatfield, 2152
 flush, 2126
 fmtflags, 2117
 getloc, 2127
 good, 2127
 goodbit, 2152
 hex, 2153
 imbue, 2127
 in, 2153
 init, 2128
 int_type, 2118
 internal, 2153
 iostate, 2118
 iword, 2128
 left, 2154
 narrow, 2129
 oct, 2154
 off_type, 2118
 openmode, 2118
 operator void *, 2129
 operator<<, 2130–2139
 out, 2154
 pos_type, 2119
 precision, 2139
 put, 2140
 pword, 2140
 rdbuf, 2141
 rdstate, 2141
 register_callback, 2142
 right, 2154
 scientific, 2154
 seekdir, 2119
 seekp, 2142, 2143
 setf, 2143
 setstate, 2144
 showbase, 2154
 showpoint, 2155
 showpos, 2155
 skipws, 2155
 str, 2144, 2145
 sync_with_stdio, 2145
 tellp, 2145
 tie, 2146
 traits_type, 2119
 trunc, 2155
 unitbuf, 2155
 unsetf, 2147
 uppercase, 2155
 widen, 2147
 width, 2147, 2148
 write, 2148
 xalloc, 2149
 std::basic_regex, 2157
 ~basic_regex, 2161
 assign, 2162–2164
 basic_regex, 2159–2161
 flags, 2164
 getloc, 2165
 imbue, 2165
 mark_count, 2165
 operator=, 2165, 2166
 swap, 2166
 std::basic_streambuf, 2168
 ~basic_streambuf, 2173
 _M_buf_locale, 2190
 _M_in_beg, 2191
 _M_in_cur, 2191
 _M_in_end, 2191
 _M_out_beg, 2191

- [_M_out_cur](#), 2192
- [_M_out_end](#), 2192
- [__streambuf_type](#), 2172
- [basic_streambuf](#), 2173
- [char_type](#), 2172
- [eback](#), 2174
- [egptr](#), 2174
- [eptr](#), 2175
- [gbump](#), 2175
- [getloc](#), 2175
- [gptr](#), 2176
- [imbue](#), 2176
- [in_avail](#), 2177
- [int_type](#), 2172
- [off_type](#), 2172
- [overflow](#), 2177
- [pbackfail](#), 2178
- [pbase](#), 2178
- [pbump](#), 2179
- [pos_type](#), 2173
- [pptr](#), 2179
- [pubimbue](#), 2180
- [pubseekoff](#), 2180
- [pubseekpos](#), 2180
- [pubsetbuf](#), 2181
- [pubsync](#), 2181
- [sbumpc](#), 2181
- [seekoff](#), 2182
- [seekpos](#), 2182
- [setbuf](#), 2182
- [setg](#), 2183
- [setp](#), 2183
- [sgetc](#), 2184
- [sgetn](#), 2184
- [showmanyc](#), 2184
- [snextc](#), 2185
- [sputbackc](#), 2185
- [sputc](#), 2186
- [sputn](#), 2186
- [stossc](#), 2187
- [sungetc](#), 2187
- [sync](#), 2187
- [traits_type](#), 2173
- [uflow](#), 2188
- [underflow](#), 2188
- [xsgetn](#), 2189
- [xspn](#), 2190
- [std::basic_string](#), 2193
 - [~basic_string](#), 2201
 - [append](#), 2202–2205
 - [assign](#), 2205–2208
 - [at](#), 2209
 - [basic_string](#), 2198–2201
 - [begin](#), 2210
 - [c_str](#), 2210
 - [capacity](#), 2210
 - [cbegin](#), 2211
 - [cend](#), 2211
 - [clear](#), 2211
 - [compare](#), 2211–2215
 - [copy](#), 2215
 - [cbegin](#), 2216
 - [crend](#), 2216
 - [data](#), 2216
 - [empty](#), 2217
 - [end](#), 2217
 - [erase](#), 2217, 2218
 - [find](#), 2219, 2220
 - [find_first_not_of](#), 2221, 2222
 - [find_first_of](#), 2223, 2224
 - [find_last_not_of](#), 2224–2226
 - [find_last_of](#), 2226–2228
 - [get_allocator](#), 2228
 - [insert](#), 2229–2233
 - [length](#), 2233
 - [max_size](#), 2234
 - [npos](#), 2251
 - [operator+=](#), 2234, 2235
 - [operator=](#), 2235–2237
 - [push_back](#), 2238
 - [rbegin](#), 2238
 - [rend](#), 2239
 - [replace](#), 2239–2245
 - [reserve](#), 2246
 - [resize](#), 2247
 - [rfind](#), 2248, 2249
 - [shrink_to_fit](#), 2249
 - [size](#), 2250
 - [substr](#), 2250
 - [swap](#), 2251
- [std::basic_stringbuf](#), 2252
 - [_M_buf_locale](#), 2275

-
- [_M_in_beg](#), 2275
 - [_M_in_cur](#), 2275
 - [_M_in_end](#), 2276
 - [_M_mode](#), 2276
 - [_M_out_beg](#), 2276
 - [_M_out_cur](#), 2276
 - [_M_out_end](#), 2277
 - [__streambuf_type](#), 2255
 - [basic_stringbuf](#), 2257
 - [char_type](#), 2255
 - [eback](#), 2257
 - [egptr](#), 2258
 - [epptr](#), 2258
 - [gbump](#), 2259
 - [getloc](#), 2259
 - [gptr](#), 2259
 - [imbue](#), 2260
 - [in_avail](#), 2260
 - [int_type](#), 2255
 - [off_type](#), 2256
 - [overflow](#), 2261
 - [pbackfail](#), 2261
 - [pbase](#), 2262
 - [pbump](#), 2262
 - [pos_type](#), 2256
 - [pptr](#), 2263
 - [pubimbue](#), 2263
 - [pubseekoff](#), 2264
 - [pubseekpos](#), 2264
 - [pubsetbuf](#), 2264
 - [pubsync](#), 2265
 - [sbumpc](#), 2265
 - [seekoff](#), 2265
 - [seekpos](#), 2266
 - [setbuf](#), 2266
 - [setg](#), 2267
 - [setp](#), 2267
 - [sgetc](#), 2268
 - [sgetn](#), 2268
 - [showmanyc](#), 2268
 - [snextc](#), 2269
 - [sputbackc](#), 2269
 - [sputc](#), 2270
 - [sputn](#), 2270
 - [stossc](#), 2271
 - [str](#), 2271
 - [sungetc](#), 2271
 - [sync](#), 2272
 - [traits_type](#), 2256
 - [uflow](#), 2272
 - [underflow](#), 2273
 - [xsgetn](#), 2273
 - [xsputn](#), 2274
 - [std::basic_stringstream](#), 2278
 - [~basic_stringstream](#), 2291
 - [_M_gcount](#), 2340
 - [_M_getloc](#), 2292
 - [_M_write](#), 2292
 - [__ctype_type](#), 2286
 - [__num_get_type](#), 2286
 - [__num_put_type](#), 2286
 - [adjustfield](#), 2341
 - [app](#), 2341
 - [ate](#), 2341
 - [bad](#), 2292
 - [badbit](#), 2341
 - [basefield](#), 2341
 - [basic_stringstream](#), 2291
 - [beg](#), 2342
 - [binary](#), 2342
 - [boolalpha](#), 2342
 - [char_type](#), 2287
 - [clear](#), 2293
 - [copyfmt](#), 2293
 - [cur](#), 2342
 - [dec](#), 2342
 - [end](#), 2343
 - [eof](#), 2293
 - [eofbit](#), 2343
 - [event](#), 2290
 - [event_callback](#), 2287
 - [exceptions](#), 2294
 - [fail](#), 2295
 - [failbit](#), 2343
 - [fill](#), 2295, 2296
 - [fixed](#), 2343
 - [flags](#), 2296
 - [floatfield](#), 2344
 - [flush](#), 2297
 - [fmtflags](#), 2287
 - [gcount](#), 2297
 - [get](#), 2297–2300
-

getline, 2301
getloc, 2302
good, 2302
goodbit, 2344
hex, 2344
ignore, 2303, 2304
imbue, 2304
in, 2344
init, 2305
int_type, 2288
internal, 2345
iostate, 2288
iword, 2305
left, 2345
narrow, 2306
oct, 2345
off_type, 2289
openmode, 2289
operator void *, 2306
operator<<, 2306–2315
operator>>, 2316–2325
out, 2345
peek, 2325
pos_type, 2289
precision, 2325
put, 2326
putback, 2326
pword, 2327
rdbuf, 2327, 2328
rdstate, 2328
read, 2329
readsome, 2329
register_callback, 2330
right, 2345
scientific, 2346
seekdir, 2290
seekg, 2330, 2331
seekp, 2332
setf, 2333
setstate, 2333
showbase, 2346
showpoint, 2346
showpos, 2346
skipws, 2346
str, 2334
sync, 2335
sync_with_stdio, 2335
tellg, 2335
tellp, 2336
tie, 2336, 2337
traits_type, 2290
trunc, 2346
unget, 2337
unitbuf, 2346
unsetf, 2338
uppercase, 2347
widen, 2338
width, 2338, 2339
write, 2339
xalloc, 2340
std::bernoulli_distribution, 2348
bernoulli_distribution, 2349
max, 2349
min, 2349
operator(), 2349
p, 2349
param, 2349, 2350
reset, 2350
result_type, 2348
std::bernoulli_distribution::param_type,
2351
std::bidirectional_iterator_tag, 2352
std::binary_function, 2353
first_argument_type, 2354
result_type, 2354
second_argument_type, 2354
std::binary_negate, 2355
first_argument_type, 2355
result_type, 2355
second_argument_type, 2356
std::binder1st, 2357
argument_type, 2358
result_type, 2358
std::binder2nd, 2359
argument_type, 2360
result_type, 2360
std::binomial_distribution, 2361
max, 2362
min, 2362
operator<<, 2364
operator>>, 2364
operator(), 2362

- p, 2363
- param, 2363
- reset, 2363
- result_type, 2362
- t, 2364
- std::binomial_distribution::param_type, 2366
- std::bitset, 2367
 - _Unchecked_flip, 2373
 - _Unchecked_reset, 2373
 - _Unchecked_set, 2373
 - _Unchecked_test, 2373
 - all, 2373
 - any, 2373
 - bitset, 2371, 2372
 - count, 2374
 - flip, 2374
 - none, 2374
 - operator<<, 2375
 - operator<<=, 2375
 - operator>>, 2376
 - operator>>=, 2376
 - operator~, 2377
 - operator^=, 2377
 - operator==, 2375
 - operator&=, 2375
 - reset, 2377, 2378
 - set, 2378
 - size, 2378
 - test, 2378
 - to_string, 2379
 - to_ulong, 2379
- std::bitset::reference, 2380
- std::cauchy_distribution, 2381
 - max, 2382
 - min, 2382
 - param, 2382
 - reset, 2382
 - result_type, 2382
- std::cauchy_distribution::param_type, 2384
- std::char_traits, 2385
- std::char_traits< __gnu_cxx::character< V, I, S > >, 2387
- std::char_traits< char >, 2388
- std::char_traits< wchar_t >, 2389
- std::chi_squared_distribution, 2390
 - max, 2391
 - min, 2391
 - operator<<, 2392
 - operator>>, 2392
 - param, 2391, 2392
 - reset, 2392
 - result_type, 2391
- std::chi_squared_distribution::param_type, 2394
- std::chrono, 727
 - duration_cast, 731
 - hours, 730
 - microseconds, 730
 - milliseconds, 730
 - minutes, 730
 - nanoseconds, 730
 - seconds, 730
 - time_point_cast, 731
- std::chrono::duration, 2395
- std::chrono::duration_values, 2397
- std::chrono::system_clock, 2398
- std::chrono::time_point, 2399
- std::chrono::treat_as_floating_point, 2400
- std::codecvt, 2401
 - do_out, 2403
 - in, 2403
 - out, 2404
 - unshift, 2405
- std::codecvt< _InternT, _ExternT, encoding_state >, 2407
 - do_out, 2409
 - in, 2409
 - out, 2410
 - unshift, 2411
- std::codecvt< char, char, mbstate_t >, 2413
 - do_out, 2415
 - in, 2415
 - out, 2416
 - unshift, 2417
- std::codecvt< wchar_t, char, mbstate_t >, 2419
 - do_out, 2421
 - in, 2421
 - out, 2422

- unshift, 2423
- std::codecvt_base, 2425
- std::codecvt_byname, 2426
 - do_out, 2428
 - in, 2428
 - out, 2429
 - unshift, 2430
- std::collate, 2432
 - ~collate, 2435
 - char_type, 2434
 - collate, 2434
 - compare, 2435
 - do_compare, 2435
 - do_hash, 2436
 - do_transform, 2436
 - hash, 2437
 - id, 2438
 - string_type, 2434
 - transform, 2437
- std::collate_byname, 2439
 - char_type, 2441
 - compare, 2441
 - do_compare, 2441
 - do_hash, 2442
 - do_transform, 2442
 - hash, 2443
 - id, 2444
 - string_type, 2441
 - transform, 2443
- std::complex, 2445
 - complex, 2446
 - operator+=, 2446
 - operator-=, 2446
 - value_type, 2446
- std::condition_variable, 2448
- std::condition_variable_any, 2449
- std::conditional, 2450
- std::const_mem_fun1_ref_t, 2451
 - first_argument_type, 2452
 - result_type, 2452
 - second_argument_type, 2452
- std::const_mem_fun1_t, 2453
 - first_argument_type, 2454
 - result_type, 2454
 - second_argument_type, 2454
- std::const_mem_fun_ref_t, 2455
 - argument_type, 2456
 - result_type, 2456
- std::const_mem_fun_t, 2457
 - argument_type, 2458
 - result_type, 2458
- std::ctype, 2459
 - char_type, 2462
 - do_is, 2462
 - do_narrow, 2463
 - do_scan_is, 2464
 - do_scan_not, 2464
 - do_tolower, 2465
 - do_toupper, 2465, 2466
 - do_widen, 2466, 2467
 - id, 2473
 - is, 2467, 2468
 - narrow, 2468, 2469
 - scan_is, 2469
 - scan_not, 2470
 - tolower, 2470
 - toupper, 2471
 - widen, 2472
- std::ctype< char >, 2474
 - ~ctype, 2477
 - char_type, 2476
 - classic_table, 2477
 - ctype, 2477
 - do_narrow, 2477, 2478
 - do_tolower, 2478, 2479
 - do_toupper, 2479, 2480
 - do_widen, 2480
 - id, 2486
 - is, 2481
 - narrow, 2482
 - scan_is, 2483
 - scan_not, 2483
 - table, 2483
 - table_size, 2486
 - tolower, 2484
 - toupper, 2484, 2485
 - widen, 2485, 2486
- std::ctype< wchar_t >, 2488
 - ~ctype, 2491
 - char_type, 2491
 - ctype, 2491
 - do_is, 2492

- do_narrow, 2492, 2493
- do_scan_is, 2493
- do_scan_not, 2494
- do_tolower, 2494, 2495
- do_toupper, 2495, 2496
- do_widen, 2496, 2497
- id, 2502
- is, 2497, 2498
- narrow, 2498
- scan_is, 2499
- scan_not, 2499
- tolower, 2500
- toupper, 2500, 2501
- widen, 2501, 2502
- std::ctype_base, 2503
- std::ctype_byname, 2504
 - char_type, 2506
 - do_is, 2507
 - do_narrow, 2507, 2508
 - do_scan_is, 2508
 - do_scan_not, 2509
 - do_tolower, 2509, 2510
 - do_toupper, 2510, 2511
 - do_widen, 2511, 2512
 - id, 2518
 - is, 2512, 2513
 - narrow, 2513, 2514
 - scan_is, 2514
 - scan_not, 2515
 - tolower, 2515
 - toupper, 2516
 - widen, 2517
- std::ctype_byname< char >, 2519
 - char_type, 2521
 - classic_table, 2522
 - do_narrow, 2522
 - do_tolower, 2523
 - do_toupper, 2523, 2524
 - do_widen, 2524, 2525
 - id, 2531
 - is, 2525, 2526
 - narrow, 2526
 - scan_is, 2527
 - scan_not, 2527
 - table, 2528
 - table_size, 2531
 - tolower, 2528
 - toupper, 2529
 - widen, 2530
- std::decay, 2532
- std::decimal, 732
 - decimal32_to_long_long, 742
- std::decimal::decimal128, 2533
 - decimal128, 2534
- std::decimal::decimal32, 2535
 - decimal32, 2536
- std::decimal::decimal64, 2537
 - decimal64, 2538
- std::default_delete, 2539
- std::defer_lock_t, 2541
- std::deque, 2542
 - ~deque, 2550
 - _M_fill_initialize, 2550
 - _M_initialize_map, 2551
 - _M_new_elements_at_back, 2551
 - _M_new_elements_at_front, 2552
 - _M_pop_back_aux, 2552
 - _M_pop_front_aux, 2552
 - _M_push_back_aux, 2552
 - _M_push_front_aux, 2552
 - _M_range_check, 2553
 - _M_range_initialize, 2553
 - _M_reallocate_map, 2554
 - _M_reserve_elements_at_back, 2554
 - _M_reserve_elements_at_front, 2554
 - _M_reserve_map_at_back, 2554
 - _M_reserve_map_at_front, 2555
- assign, 2555, 2556
- at, 2556, 2557
- back, 2557
- begin, 2558
- cbegin, 2558
- cend, 2558
- clear, 2558
- crbegin, 2559
- crend, 2559
- deque, 2548–2550
- emplace, 2559
- empty, 2560
- end, 2560

- erase, 2560, 2561
- front, 2561
- get_allocator, 2562
- insert, 2562, 2563
- max_size, 2564
- operator=, 2564, 2565
- pop_back, 2566
- pop_front, 2566
- push_back, 2567
- push_front, 2567
- rbegin, 2567, 2568
- rend, 2568
- resize, 2568
- shrink_to_fit, 2569
- size, 2569
- swap, 2569
- std::discard_block_engine, 2571
 - base, 2574
 - discard, 2574
 - discard_block_engine, 2572, 2573
 - max, 2574
 - min, 2575
 - operator<<, 2576
 - operator>>, 2577
 - operator(), 2575
 - operator==, 2576
 - result_type, 2572
 - seed, 2575, 2576
- std::discrete_distribution, 2578
 - max, 2579
 - min, 2579
 - operator<<, 2580
 - operator>>, 2580
 - param, 2579, 2580
 - probabilities, 2580
 - reset, 2580
 - result_type, 2579
- std::discrete_distribution::param_type, 2582
- std::divides, 2583
 - first_argument_type, 2584
 - result_type, 2584
 - second_argument_type, 2584
- std::domain_error, 2585
 - what, 2585
- std::enable_if, 2587
- std::enable_shared_from_this, 2588
- std::equal_to, 2589
 - first_argument_type, 2590
 - result_type, 2590
 - second_argument_type, 2590
- std::error_category, 2591
- std::error_code, 2592
- std::error_condition, 2593
- std::exception, 2595
 - what, 2596
- std::exponential_distribution, 2597
 - exponential_distribution, 2598
 - lambda, 2598
 - max, 2598
 - min, 2598
 - param, 2599
 - reset, 2599
 - result_type, 2598
- std::exponential_distribution::param_type, 2600
- std::extent, 2601
- std::extreme_value_distribution, 2602
 - a, 2603
 - b, 2603
 - max, 2603
 - min, 2603
 - param, 2603, 2604
 - reset, 2604
 - result_type, 2603
- std::extreme_value_distribution::param_type, 2605
- std::fisher_f_distribution, 2606
 - max, 2607
 - min, 2607
 - operator<<, 2608
 - operator>>, 2608
 - param, 2607, 2608
 - reset, 2608
 - result_type, 2607
- std::fisher_f_distribution::param_type, 2610
- std::forward_iterator_tag, 2611
- std::forward_list, 2612
 - ~forward_list, 2618
 - assign, 2618, 2619
 - before_begin, 2619

- begin, 2620
- cbefore_begin, 2620
- cbegin, 2620
- cend, 2621
- clear, 2621
- emplace_after, 2621
- emplace_front, 2622
- empty, 2622
- end, 2622
- erase_after, 2623
- forward_list, 2615–2618
- front, 2624
- get_allocator, 2624
- insert_after, 2624–2626
- max_size, 2626
- merge, 2626, 2627
- operator=, 2627, 2628
- pop_front, 2628
- push_front, 2628
- remove, 2629
- remove_if, 2629
- resize, 2629, 2630
- reverse, 2630
- sort, 2630, 2631
- splice_after, 2631, 2632
- swap, 2632
- unique, 2632, 2633
- std::fpos, 2634
 - fpos, 2634
 - operator streamoff, 2635
 - operator+, 2635
 - operator+=, 2635
 - operator-, 2635
 - operator-=, 2635
 - state, 2635, 2636
- std::front_insert_iterator, 2637
 - container_type, 2638
 - difference_type, 2638
 - front_insert_iterator, 2639
 - iterator_category, 2638
 - operator*, 2639
 - operator++, 2639
 - operator=, 2640
 - pointer, 2638
 - reference, 2638
 - value_type, 2639
- std::function< _Res(_ArgTypes...)>, 2641
 - function, 2643, 2644
 - operator bool, 2644
 - operator(), 2645
 - operator=, 2645, 2646
 - swap, 2647
 - target, 2647
 - target_type, 2648
- std::future, 2649
 - _M_get_result, 2650
 - future, 2650
 - get, 2650
- std::future< _Res & >, 2652
 - _M_get_result, 2653
 - future, 2653
 - get, 2653
- std::future< void >, 2655
 - _M_get_result, 2656
 - future, 2656
 - get, 2656
- std::future_error, 2658
 - what, 2658
- std::gamma_distribution, 2660
 - alpha, 2662
 - beta, 2662
 - gamma_distribution, 2661
 - max, 2662
 - min, 2662
 - operator<<, 2663
 - operator>>, 2664
 - operator(), 2662
 - param, 2662, 2663
 - reset, 2663
 - result_type, 2661
- std::gamma_distribution::param_type, 2665
- std::geometric_distribution, 2666
 - max, 2667
 - min, 2667
 - p, 2667
 - param, 2667
 - reset, 2668
 - result_type, 2667
- std::geometric_distribution::param_type, 2669

-
- std::greater, 2670
 - first_argument_type, 2671
 - result_type, 2671
 - second_argument_type, 2671
 - std::greater_equal, 2672
 - first_argument_type, 2673
 - result_type, 2673
 - second_argument_type, 2673
 - std::gslice, 2674
 - std::gslice_array, 2675
 - std::has_nothrow_assign, 2677
 - std::has_nothrow_copy_constructor, 2678
 - std::has_nothrow_default_constructor, 2679
 - std::has_trivial_assign, 2680
 - std::has_trivial_copy_constructor, 2681
 - std::has_trivial_default_constructor, 2682
 - std::has_trivial_destructor, 2683
 - std::has_virtual_destructor, 2684
 - std::hash, 2685
 - argument_type, 2686
 - result_type, 2686
 - std::hash< ::bitset< _Nb > >, 2688
 - argument_type, 2689
 - result_type, 2689
 - std::hash< ::vector< bool, _Alloc > >, 2690
 - argument_type, 2691
 - result_type, 2691
 - std::hash< __debug::bitset< _Nb > >, 2692
 - argument_type, 2693
 - result_type, 2693
 - std::hash< __debug::vector< bool, _Alloc > >, 2694
 - argument_type, 2695
 - result_type, 2695
 - std::hash< __gnu_cxx::throw_value_-limit >, 2696
 - argument_type, 2697
 - result_type, 2697
 - std::hash< __gnu_cxx::throw_value_-random >, 2698
 - argument_type, 2699
 - result_type, 2699
 - std::hash< __profile::bitset< _Nb > >, 2700
 - argument_type, 2701
 - result_type, 2701
 - std::hash< __profile::vector< bool, _Alloc > >, 2702
 - argument_type, 2703
 - result_type, 2703
 - std::hash< _Tp * >, 2704
 - argument_type, 2705
 - result_type, 2705
 - std::hash< error_code >, 2706
 - argument_type, 2707
 - result_type, 2707
 - std::hash< string >, 2708
 - argument_type, 2709
 - result_type, 2709
 - std::hash< thread::id >, 2710
 - argument_type, 2711
 - result_type, 2711
 - std::hash< u16string >, 2712
 - argument_type, 2713
 - result_type, 2713
 - std::hash< u32string >, 2714
 - argument_type, 2715
 - result_type, 2715
 - std::hash< wstring >, 2716
 - argument_type, 2717
 - result_type, 2717
 - std::identity, 2718
 - std::independent_bits_engine, 2719
 - base, 2722
 - discard, 2722
 - independent_bits_engine, 2720, 2721
 - max, 2722
 - min, 2722
 - operator>>, 2724
 - operator(), 2723
 - operator==, 2724
 - result_type, 2720
 - seed, 2723
 - std::indirect_array, 2725
 - std::initializer_list, 2727
 - std::input_iterator_tag, 2728
 - std::insert_iterator, 2729
-

- container_type, 2730
- difference_type, 2730
- insert_iterator, 2731
- iterator_category, 2730
- operator*, 2731
- operator++, 2731
- operator=, 2732
- pointer, 2730
- reference, 2730
- value_type, 2731
- std::integral_constant, 2733
- std::invalid_argument, 2734
 - what, 2734
- std::ios_base, 2736
 - ~ios_base, 2741
 - _M_getloc, 2741
 - adjustfield, 2747
 - app, 2748
 - ate, 2748
 - badbit, 2748
 - basefield, 2748
 - beg, 2748
 - binary, 2749
 - boolalpha, 2749
 - cur, 2749
 - dec, 2749
 - end, 2749
 - eofbit, 2750
 - event, 2741
 - event_callback, 2739
 - failbit, 2750
 - fixed, 2750
 - flags, 2742
 - floatfield, 2750
 - fmtflags, 2739
 - getloc, 2742
 - goodbit, 2751
 - hex, 2751
 - imbue, 2743
 - in, 2751
 - internal, 2751
 - iostate, 2740
 - isword, 2743
 - left, 2752
 - oct, 2752
 - openmode, 2740
 - out, 2752
 - precision, 2743, 2744
 - pword, 2744
 - register_callback, 2744
 - right, 2752
 - scientific, 2752
 - seekdir, 2741
 - setf, 2745
 - showbase, 2753
 - showpoint, 2753
 - showpos, 2753
 - skipws, 2753
 - sync_with_stdio, 2746
 - trunc, 2753
 - unitbuf, 2753
 - unsetf, 2746
 - uppercase, 2753
 - width, 2746, 2747
 - xalloc, 2747
- std::ios_base::failure, 2755
 - what, 2755
- std::is_abstract, 2756
- std::is_arithmetic, 2757
- std::is_array, 2758
- std::is_base_of, 2759
- std::is_bind_expression, 2760
- std::is_bind_expression< _Bind< _-
Signature > >, 2762
- std::is_bind_expression< _Bind_result<
_Result, _Signature > >, 2763
- std::is_class, 2764
- std::is_compound, 2765
- std::is_const, 2766
- std::is_constructible, 2767
- std::is_convertible, 2768
- std::is_empty, 2769
- std::is_enum, 2770
- std::is_error_code_enum, 2771
- std::is_error_condition_enum, 2772
- std::is_explicitly_convertible, 2773
- std::is_floating_point, 2774
- std::is_function, 2775
- std::is_fundamental, 2776
- std::is_integral, 2777
- std::is_lvalue_reference, 2778
- std::is_member_function_pointer, 2779

- std::is_member_object_pointer, 2780
- std::is_object, 2781
- std::is_placeholder, 2782
- std::is_placeholder< _Placeholder< _-
Num > >, 2784
- std::is_pod, 2785
- std::is_pointer, 2786
- std::is_polymorphic, 2787
- std::is_reference, 2788
- std::is_rvalue_reference, 2789
- std::is_same, 2790
- std::is_scalar, 2791
- std::is_signed, 2792
- std::is_standard_layout, 2793
- std::is_trivial, 2794
- std::is_union, 2795
- std::is_unsigned, 2796
- std::is_void, 2797
- std::is_volatile, 2798
- std::istream_iterator, 2799
 - difference_type, 2800
 - istream_iterator, 2801
 - iterator_category, 2800
 - pointer, 2800
 - reference, 2800
 - value_type, 2800
- std::istreambuf_iterator, 2802
 - char_type, 2803
 - difference_type, 2803
 - equal, 2805
 - int_type, 2803
 - istream_type, 2803
 - istreambuf_iterator, 2805
 - iterator_category, 2804
 - operator*, 2805
 - operator++, 2806
 - pointer, 2804
 - reference, 2804
 - streambuf_type, 2804
 - traits_type, 2804
 - value_type, 2804
- std::iterator, 2807
 - difference_type, 2807
 - iterator_category, 2808
 - pointer, 2808
 - reference, 2808
 - value_type, 2808
- std::iterator_traits, 2809
 - std::iterator_traits< _Tp * >, 2810
 - std::iterator_traits< const _Tp * >, 2811
- std::length_error, 2812
 - what, 2812
- std::less, 2814
 - first_argument_type, 2815
 - result_type, 2815
 - second_argument_type, 2815
- std::less_equal, 2816
 - first_argument_type, 2817
 - result_type, 2817
 - second_argument_type, 2817
- std::linear_congruential_engine, 2818
 - discard, 2820
 - increment, 2824
 - linear_congruential_engine, 2819,
2820
 - max, 2820
 - min, 2821
 - modulus, 2824
 - multiplier, 2824
 - operator<<, 2822
 - operator>>, 2823
 - operator(), 2821
 - operator==, 2822
 - result_type, 2819
 - seed, 2821
- std::list, 2825
 - _M_create_node, 2831
 - assign, 2832
 - back, 2833
 - begin, 2833
 - cbegin, 2833
 - cend, 2834
 - clear, 2834
 - crbegin, 2834
 - crend, 2834
 - emplace, 2834
 - empty, 2835
 - end, 2835
 - erase, 2836
 - front, 2837
 - get_allocator, 2837
 - insert, 2837–2839

- list, 2829–2831
- max_size, 2839
- merge, 2840
- operator=, 2840, 2841
- pop_back, 2842
- pop_front, 2842
- push_back, 2842
- push_front, 2842
- rbegin, 2843
- remove, 2843
- remove_if, 2844
- rend, 2844
- resize, 2844
- reverse, 2845
- size, 2845
- sort, 2845
- splice, 2846
- swap, 2847
- unique, 2847, 2848
- std::locale, 2849
 - ~locale, 2852
 - all, 2855
 - category, 2851
 - classic, 2853
 - collate, 2855
 - combine, 2853
 - ctype, 2855
 - global, 2853
 - locale, 2851, 2852
 - messages, 2856
 - monetary, 2856
 - name, 2853
 - none, 2856
 - numeric, 2856
 - operator(), 2854
 - operator=, 2854
 - operator==, 2855
 - time, 2857
- std::locale::facet, 2858
 - ~facet, 2859
 - facet, 2859
- std::locale::id, 2860
 - has_facet, 2861
 - id, 2860
 - use_facet, 2861
- std::lock_guard, 2862
- std::logic_error, 2863
 - logic_error, 2863
 - what, 2864
- std::logical_and, 2865
 - first_argument_type, 2866
 - result_type, 2866
 - second_argument_type, 2866
- std::logical_not, 2867
 - argument_type, 2868
 - result_type, 2868
- std::logical_or, 2869
 - first_argument_type, 2870
 - result_type, 2870
 - second_argument_type, 2870
- std::lognormal_distribution, 2871
 - max, 2872
 - min, 2872
 - operator<<, 2873
 - operator>>, 2873
 - param, 2872, 2873
 - reset, 2873
 - result_type, 2872
- std::lognormal_distribution::param_type, 2875
- std::make_signed, 2876
- std::make_unsigned, 2877
- std::map, 2878
 - at, 2883
 - begin, 2883
 - cbegin, 2884
 - cend, 2884
 - clear, 2884
 - count, 2884
 - crbegin, 2885
 - crend, 2885
 - empty, 2885
 - end, 2886
 - equal_range, 2886, 2887
 - erase, 2887, 2888
 - find, 2888, 2889
 - get_allocator, 2889
 - insert, 2890, 2891
 - key_comp, 2891
 - lower_bound, 2892
 - map, 2880–2882
 - max_size, 2892

- operator=, 2893
- rbegin, 2894, 2895
- rend, 2895
- size, 2895
- swap, 2896
- upper_bound, 2896
- value_comp, 2897
- std::mask_array, 2898
- std::match_results, 2900
 - ~match_results, 2904
 - begin, 2905
 - cbegin, 2905
 - cend, 2905
 - empty, 2905
 - end, 2906
 - format, 2906
 - length, 2906
 - match_results, 2904
 - operator=, 2907
 - position, 2907
 - prefix, 2908
 - size, 2908
 - str, 2908
 - suffix, 2909
 - swap, 2909
- std::mem_fun1_ref_t, 2910
 - first_argument_type, 2911
 - result_type, 2911
 - second_argument_type, 2911
- std::mem_fun1_t, 2912
 - first_argument_type, 2913
 - result_type, 2913
 - second_argument_type, 2913
- std::mem_fun_ref_t, 2914
 - argument_type, 2915
 - result_type, 2915
- std::mem_fun_t, 2916
 - argument_type, 2917
 - result_type, 2917
- std::messages, 2918
 - ~messages, 2921
 - char_type, 2920
 - id, 2921
 - messages, 2920, 2921
 - string_type, 2920
- std::messages_base, 2922
 - std::messages_byname, 2923
 - char_type, 2925
 - id, 2925
 - string_type, 2925
 - std::minus, 2926
 - first_argument_type, 2927
 - result_type, 2927
 - second_argument_type, 2927
 - std::modulus, 2928
 - first_argument_type, 2929
 - result_type, 2929
 - second_argument_type, 2929
 - std::money_base, 2930
 - std::money_get, 2932
 - ~money_get, 2934
 - char_type, 2933
 - do_get, 2935
 - get, 2935, 2936
 - id, 2937
 - iter_type, 2933
 - money_get, 2934
 - string_type, 2934
 - std::money_put, 2938
 - ~money_put, 2940
 - char_type, 2939
 - do_put, 2941
 - id, 2943
 - iter_type, 2939
 - money_put, 2940
 - put, 2942
 - string_type, 2940
 - std::moneypunct, 2944
 - ~moneypunct, 2948
 - char_type, 2946
 - curr_symbol, 2948
 - decimal_point, 2948
 - do_curr_symbol, 2948
 - do_decimal_point, 2949
 - do_frac_digits, 2949
 - do_grouping, 2950
 - do_neg_format, 2950
 - do_negative_sign, 2950
 - do_pos_format, 2951
 - do_positive_sign, 2951
 - do_thousands_sep, 2952
 - frac_digits, 2952

- grouping, 2952
- id, 2955
- intl, 2955
- money_punct, 2947
- neg_format, 2953
- negative_sign, 2953
- pos_format, 2954
- positive_sign, 2954
- string_type, 2946
- thousands_sep, 2955
- std::money_punct_byname, 2957
- char_type, 2959
- curr_symbol, 2960
- decimal_point, 2960
- do_curr_symbol, 2960
- do_decimal_point, 2961
- do_frac_digits, 2961
- do_grouping, 2961
- do_neg_format, 2962
- do_negative_sign, 2962
- do_pos_format, 2962
- do_positive_sign, 2963
- do_thousands_sep, 2963
- frac_digits, 2964
- grouping, 2964
- id, 2967
- intl, 2967
- neg_format, 2964
- negative_sign, 2965
- pos_format, 2965
- positive_sign, 2966
- string_type, 2959
- thousands_sep, 2966
- std::move_iterator, 2968
- std::multimap, 2970
 - begin, 2975
 - cbegin, 2975
 - cend, 2975
 - clear, 2976
 - count, 2976
 - crbegin, 2976
 - crend, 2976
 - empty, 2977
 - end, 2977
 - equal_range, 2977, 2978
 - erase, 2978, 2979
 - find, 2980
 - get_allocator, 2981
 - insert, 2981, 2982
 - key_comp, 2983
 - lower_bound, 2983, 2984
 - max_size, 2984
 - multimap, 2972–2974
 - operator=, 2984, 2985
 - rbegin, 2985, 2986
 - rend, 2986
 - size, 2986
 - swap, 2987
 - upper_bound, 2987
 - value_comp, 2988
- std::multiplies, 2989
 - first_argument_type, 2990
 - result_type, 2990
 - second_argument_type, 2990
- std::multiset, 2991
 - begin, 2996
 - cbegin, 2996
 - cend, 2996
 - clear, 2996
 - count, 2996
 - crbegin, 2997
 - crend, 2997
 - empty, 2997
 - end, 2997
 - equal_range, 2997, 2998
 - erase, 2998, 2999
 - find, 3000
 - get_allocator, 3001
 - insert, 3001, 3002
 - key_comp, 3003
 - lower_bound, 3003, 3004
 - max_size, 3004
 - multiset, 2993–2995
 - operator<, 3008
 - operator=, 3004, 3005
 - operator==, 3008
 - rbegin, 3005
 - rend, 3006
 - size, 3006
 - swap, 3006
 - upper_bound, 3006, 3007
 - value_comp, 3007

- std::mutex, 3010
- std::negate, 3011
 - argument_type, 3012
 - result_type, 3012
- std::negative_binomial_distribution, 3013
 - k, 3014
 - max, 3014
 - min, 3014
 - operator<<, 3016
 - operator>>, 3016
 - p, 3014
 - param, 3015
 - reset, 3015
 - result_type, 3014
- std::negative_binomial_-distribution::param_type, 3017
- std::nested_exception, 3018
- std::normal_distribution, 3019
 - max, 3021
 - mean, 3021
 - min, 3021
 - normal_distribution, 3020
 - operator<<, 3022
 - operator>>, 3023
 - operator(), 3021
 - param, 3021, 3022
 - reset, 3022
 - result_type, 3020
 - stddev, 3022
- std::normal_distribution::param_type, 3024
- std::not_equal_to, 3025
 - first_argument_type, 3026
 - result_type, 3026
 - second_argument_type, 3026
- std::num_get, 3027
 - ~num_get, 3030
 - char_type, 3029
 - do_get, 3030–3032
 - get, 3033–3037
 - id, 3038
 - iter_type, 3029
 - num_get, 3030
- std::num_put, 3039
 - ~num_put, 3042
 - char_type, 3041
 - do_put, 3042, 3043
 - id, 3048
 - iter_type, 3041
 - num_put, 3041
 - put, 3044–3047
- std::numeric_limits, 3049
 - denorm_min, 3050
 - digits, 3052
 - digits10, 3052
 - epsilon, 3050
 - has_denorm, 3052
 - has_denorm_loss, 3052
 - has_infinity, 3052
 - has_quiet_NaN, 3053
 - has_signaling_NaN, 3053
 - infinity, 3050
 - is_bounded, 3053
 - is_exact, 3053
 - is_iec559, 3053
 - is_integer, 3053
 - is_modulo, 3054
 - is_signed, 3054
 - is_specialized, 3054
 - lowest, 3051
 - max, 3051
 - max_digits10, 3054
 - max_exponent, 3054
 - max_exponent10, 3055
 - min, 3051
 - min_exponent, 3055
 - min_exponent10, 3055
 - quiet_NaN, 3051
 - radix, 3055
 - round_error, 3051
 - round_style, 3055
 - signaling_NaN, 3051
 - tinyness_before, 3055
 - traps, 3056
- std::numeric_limits< bool >, 3057
- std::numeric_limits< char >, 3059
- std::numeric_limits< char16_t >, 3061
- std::numeric_limits< char32_t >, 3063
- std::numeric_limits< double >, 3065
- std::numeric_limits< float >, 3067
- std::numeric_limits< int >, 3069

- std::numeric_limits< long >, 3071
- std::numeric_limits< long double >, 3073
- std::numeric_limits< long long >, 3075
- std::numeric_limits< short >, 3077
- std::numeric_limits< signed char >, 3079
- std::numeric_limits< unsigned char >, 3081
- std::numeric_limits< unsigned int >, 3083
- std::numeric_limits< unsigned long >, 3085
- std::numeric_limits< unsigned long long >, 3087
- std::numeric_limits< unsigned short >, 3089
- std::numeric_limits< wchar_t >, 3091
- std::numprint, 3093
 - ~numprint, 3096
 - char_type, 3095
 - decimal_point, 3096
 - do_decimal_point, 3097
 - do_falsename, 3097
 - do_grouping, 3097
 - do_thousands_sep, 3098
 - do_truename, 3098
 - falsename, 3098
 - grouping, 3099
 - id, 3100
 - numprint, 3095, 3096
 - string_type, 3095
 - thousands_sep, 3099
 - truename, 3100
- std::numprint_byname, 3101
 - char_type, 3103
 - decimal_point, 3103
 - do_decimal_point, 3103
 - do_falsename, 3104
 - do_grouping, 3104
 - do_thousands_sep, 3104
 - do_truename, 3105
 - falsename, 3105
 - grouping, 3105
 - id, 3107
 - string_type, 3103
 - thousands_sep, 3106
 - truename, 3106
- std::once_flag, 3108
- std::ostream_iterator, 3109
 - char_type, 3110
 - difference_type, 3110
 - iterator_category, 3110
 - operator=, 3112
 - ostream_iterator, 3112
 - ostream_type, 3110
 - pointer, 3111
 - reference, 3111
 - traits_type, 3111
 - value_type, 3111
- std::ostreambuf_iterator, 3114
 - char_type, 3115
 - difference_type, 3115
 - failed, 3117
 - iterator_category, 3115
 - operator*, 3117
 - operator++, 3117
 - operator=, 3118
 - ostream_type, 3115
 - ostreambuf_iterator, 3117
 - pointer, 3116
 - reference, 3116
 - streambuf_type, 3116
 - traits_type, 3116
 - value_type, 3116
- std::out_of_range, 3119
 - what, 3119
- std::output_iterator_tag, 3121
- std::overflow_error, 3122
 - what, 3122
- std::owner_less< shared_ptr< _Tp > >, 3124
- std::owner_less< weak_ptr< _Tp > >, 3125
- std::packaged_task< _Res(_ArgTypes...)>, 3126
- std::pair, 3127
 - first, 3129
 - first_type, 3128
 - pair, 3128, 3129
 - second, 3129
 - second_type, 3128

- std::piecewise_constant_distribution,
 - 3130
 - densities, 3131
 - intervals, 3131
 - max, 3132
 - min, 3132
 - operator<<, 3133
 - operator>>, 3133
 - param, 3132
 - reset, 3132
 - result_type, 3131
- std::piecewise_constant_-distribution::param_type, 3135
- std::piecewise_linear_distribution, 3136
 - densities, 3137
 - intervals, 3137
 - max, 3138
 - min, 3138
 - operator<<, 3139
 - operator>>, 3139
 - param, 3138
 - reset, 3138
 - result_type, 3137
- std::piecewise_linear_-distribution::param_type, 3141
- std::placeholders, 743
- std::plus, 3142
 - first_argument_type, 3143
 - result_type, 3143
 - second_argument_type, 3143
- std::pointer_to_binary_function, 3144
 - first_argument_type, 3145
 - result_type, 3145
 - second_argument_type, 3145
- std::pointer_to_unary_function, 3146
 - argument_type, 3147
 - result_type, 3147
- std::poisson_distribution, 3148
 - max, 3149
 - mean, 3149
 - min, 3149
 - operator<<, 3151
 - operator>>, 3151
 - operator(), 3149
 - param, 3150
 - reset, 3150
 - result_type, 3149
- std::poisson_distribution::param_type, 3152
- std::priority_queue, 3153
 - empty, 3155
 - pop, 3155
 - priority_queue, 3154
 - push, 3156
 - size, 3156
 - top, 3156
- std::promise, 3158
- std::promise<_Res & >, 3159
- std::promise<void >, 3160
- std::queue, 3161
 - back, 3163
 - c, 3164
 - empty, 3163
 - front, 3163
 - pop, 3163
 - push, 3164
 - queue, 3162
 - size, 3164
- std::random_access_iterator_tag, 3165
- std::random_device, 3166
 - result_type, 3166
- std::range_error, 3167
 - what, 3167
- std::rank, 3169
- std::ratio, 3170
- std::ratio_add, 3171
- std::ratio_divide, 3172
- std::ratio_equal, 3173
- std::ratio_greater, 3174
- std::ratio_greater_equal, 3175
- std::ratio_less, 3176
- std::ratio_less_equal, 3177
- std::ratio_multiply, 3178
- std::ratio_not_equal, 3179
- std::ratio_subtract, 3180
- std::raw_storage_iterator, 3181
 - difference_type, 3182
 - iterator_category, 3182
 - pointer, 3182
 - reference, 3182

- value_type, 3182
- std::recursive_mutex, 3184
- std::recursive_timed_mutex, 3185
- std::reference_wrapper, 3186
- std::regex_constants, 744
 - __match_flag, 746
 - __syntax_option, 746
- awk, 748
- basic, 748
- collate, 748
- ECMAScript, 749
- egrep, 749
- error_backref, 746
- error_badbrace, 746
- error_badrepeat, 747
- error_brace, 747
- error_brack, 747
- error_collate, 747
- error_complexity, 747
- error_ctype, 747
- error_escape, 747
- error_paren, 747
- error_range, 748
- error_space, 748
- error_stack, 748
- error_type, 746
- extended, 749
- format_default, 749
- format_first_only, 750
- format_no_copy, 750
- format_sed, 750
- grep, 750
- icase, 751
- match_any, 751
- match_continuous, 751
- match_default, 751
- match_flag_type, 745
- match_not_bol, 751
- match_not_bow, 751
- match_not_eol, 752
- match_not_eow, 752
- match_not_null, 752
- match_prev_avail, 752
- nosubs, 752
- optimize, 752
- syntax_option_type, 745
- std::regex_error, 3188
 - code, 3189
 - regex_error, 3189
 - what, 3189
- std::regex_iterator, 3190
 - operator*, 3192
 - operator++, 3192
 - operator->, 3193
 - operator=, 3193
 - operator==, 3193
 - regex_iterator, 3191
- std::regex_token_iterator, 3195
 - operator*, 3198
 - operator++, 3199
 - operator->, 3199
 - operator=, 3200
 - operator==, 3200
 - regex_token_iterator, 3196–3198
- std::regex_traits, 3201
 - getloc, 3202
 - imbue, 3202
 - length, 3203
 - lookup_classname, 3203
 - lookup_collatename, 3204
 - regex_traits, 3202
 - transform, 3204
 - transform_primary, 3205
 - translate, 3206
 - translate_nocase, 3206
- std::rel_ops, 754
 - operator<=, 754
 - operator>, 755
 - operator>=, 755
- std::remove_all_extents, 3207
- std::remove_const, 3208
- std::remove_cv, 3209
- std::remove_extent, 3210
- std::remove_pointer, 3211
- std::remove_reference, 3212
- std::remove_volatile, 3213
- std::reverse_iterator, 3214
 - base, 3217
 - difference_type, 3215
 - iterator_category, 3215
 - operator*, 3217
 - operator+, 3218

- operator++, 3218
- operator+=", 3218
- operator-, 3219
- operator->, 3220
- operator--, 3219
- operator=, 3220
- pointer, 3216
- reference, 3216
- reverse_iterator, 3216, 3217
- value_type, 3216
- std::runtime_error, 3222
 - runtime_error, 3222
 - what, 3223
- std::seed_seq, 3224
 - result_type, 3224
 - seed_seq, 3224
- std::set, 3226
 - allocator_type, 3228
 - begin, 3234
 - cbegin, 3234
 - end, 3234
 - clear, 3234
 - const_iterator, 3228
 - const_pointer, 3228
 - const_reference, 3229
 - const_reverse_iterator, 3229
 - count, 3234
 - crbegin, 3235
 - crend, 3235
 - difference_type, 3229
 - empty, 3235
 - end, 3235
 - equal_range, 3235, 3236
 - erase, 3236, 3237
 - find, 3238
 - get_allocator, 3238
 - insert, 3238–3240
 - iterator, 3229
 - key_comp, 3240
 - key_compare, 3229
 - key_type, 3230
 - lower_bound, 3240, 3241
 - max_size, 3241
 - operator=, 3241, 3242
 - pointer, 3230
 - rbegin, 3242
 - reference, 3230
 - rend, 3243
 - reverse_iterator, 3230
 - set, 3231–3233
 - size, 3243
 - size_type, 3230
 - swap, 3243
 - upper_bound, 3243, 3244
 - value_comp, 3244
 - value_compare, 3231
 - value_type, 3231
- std::shared_future, 3245
 - _M_get_result, 3246
 - get, 3247
 - shared_future, 3246
- std::shared_future< _Res & >, 3248
 - _M_get_result, 3250
 - get, 3250
 - shared_future, 3249
- std::shared_future< void >, 3251
 - _M_get_result, 3253
 - shared_future, 3252
- std::shared_ptr, 3254
 - allocate_shared, 3259
 - shared_ptr, 3255–3258
- std::shuffle_order_engine, 3260
 - base, 3263
 - discard, 3263
 - max, 3263
 - min, 3264
 - operator<<, 3265
 - operator>>, 3266
 - operator(), 3264
 - operator==, 3265
 - result_type, 3261
 - seed, 3264
 - shuffle_order_engine, 3261, 3262
- std::slice, 3267
- std::slice_array, 3268
- std::stack, 3270
 - empty, 3271
 - pop, 3271
 - push, 3272
 - size, 3272
 - stack, 3271
 - top, 3272

- std::student_t_distribution, 3274
 - max, 3275
 - min, 3275
 - operator<<, 3276
 - operator>>, 3276
 - param, 3275, 3276
 - reset, 3276
 - result_type, 3275
- std::student_t_distribution::param_type, 3278
- std::sub_match, 3279
 - compare, 3281
 - first, 3283
 - first_type, 3280
 - length, 3282
 - operator basic_string< value_type >, 3282
 - second, 3283
 - second_type, 3280
 - str, 3282
- std::system_error, 3284
 - what, 3285
- std::this_thread, 756
 - get_id, 756
 - sleep_for, 756
 - sleep_until, 756
 - yield, 757
- std::thread, 3286
 - native_handle, 3287
- std::thread::id, 3288
- std::time_base, 3289
- std::time_get, 3290
 - ~time_get, 3293
 - char_type, 3292
 - date_order, 3293
 - do_date_order, 3293
 - do_get_date, 3294
 - do_get_monthname, 3294
 - do_get_time, 3295
 - do_get_weekday, 3295
 - do_get_year, 3296
 - get_date, 3297
 - get_monthname, 3297
 - get_time, 3298
 - get_weekday, 3299
 - get_year, 3299
- id, 3300
- iter_type, 3292
- time_get, 3293
- std::time_get_byname, 3301
 - char_type, 3303
 - date_order, 3303
 - do_date_order, 3304
 - do_get_date, 3304
 - do_get_monthname, 3305
 - do_get_time, 3305
 - do_get_weekday, 3306
 - do_get_year, 3306
 - get_date, 3307
 - get_monthname, 3308
 - get_time, 3308
 - get_weekday, 3309
 - get_year, 3310
 - id, 3310
 - iter_type, 3303
- std::time_put, 3312
 - ~time_put, 3314
 - char_type, 3313
 - do_put, 3314
 - id, 3316
 - iter_type, 3313
 - put, 3315
 - time_put, 3314
- std::time_put_byname, 3317
 - char_type, 3318
 - do_put, 3319
 - id, 3321
 - iter_type, 3318
 - put, 3319, 3320
- std::timed_mutex, 3322
- std::tr1, 758
- std::tr1::__detail, 762
- std::try_to_lock_t, 3323
- std::tuple, 3324
- std::tuple< _T1, _T2 >, 3326
- std::tuple_element< 0, tuple< _Head, _Tail...> >, 3328
- std::tuple_element< __i, tuple< _Head, _Tail...> >, 3329
- std::tuple_size< tuple< _Elements...> >, 3330
- std::type_info, 3331

- ~type_info, 3331
- name, 3332
- std::unary_function, 3334
 - argument_type, 3335
 - result_type, 3335
- std::unary_negate, 3336
 - argument_type, 3337
 - result_type, 3337
- std::underflow_error, 3338
 - what, 3338
- std::uniform_int_distribution, 3340
 - max, 3341
 - min, 3341
 - operator(), 3342
 - param, 3342
 - reset, 3343
 - result_type, 3341
 - uniform_int_distribution, 3341
- std::uniform_int_distribution::param_type, 3344
- std::uniform_real_distribution, 3345
 - max, 3346
 - min, 3346
 - param, 3346, 3347
 - reset, 3347
 - result_type, 3346
 - uniform_real_distribution, 3346
- std::uniform_real_distribution::param_type, 3348
- std::unique_lock, 3349
- std::unique_ptr, 3351
- std::unordered_map, 3355
- std::unordered_multimap, 3357
- std::unordered_multiset, 3359
- std::unordered_set, 3361
- std::valarray, 3363
 - valarray, 3366
- std::vector, 3367
 - ~vector, 3373
 - _M_allocate_and_copy, 3373
 - _M_range_check, 3373
 - assign, 3373, 3374
 - at, 3375
 - back, 3375, 3376
 - begin, 3376
 - capacity, 3376
 - cbegin, 3377
 - cend, 3377
 - clear, 3377
 - crbegin, 3377
 - crend, 3378
 - data, 3378
 - emplace, 3378
 - empty, 3379
 - end, 3379
 - erase, 3379, 3380
 - front, 3380, 3381
 - insert, 3381, 3382
 - max_size, 3383
 - operator=, 3383, 3384
 - pop_back, 3385
 - push_back, 3385
 - rbegin, 3386
 - rend, 3386
 - reserve, 3386
 - resize, 3387
 - shrink_to_fit, 3387
 - size, 3388
 - swap, 3388
 - vector, 3371, 3372
- std::vector< bool, _Alloc >, 3389
- std::weak_ptr, 3394
- std::weibull_distribution, 3395
 - a, 3396
 - b, 3396
 - max, 3396
 - min, 3396
 - param, 3396, 3397
 - reset, 3397
 - result_type, 3396
- std::weibull_distribution::param_type, 3398
- stdatomic.h, 3871
- stddev
 - std::normal_distribution, 3022
- stdexcept, 3872
- stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 969, 970
- stdio_filebuf.h, 3873
- stdio_sync_filebuf.h, 3874
- stl_algo.h, 3875
- stl_algobase.h, 3889

- stl_bvector.h, 3892
- stl_construct.h, 3894
- stl_deque.h, 3895
 - _GLIBCXX_DEQUE_BUF_SIZE, 3898
- stl_function.h, 3899
- stl_heap.h, 3902
- stl_iterator.h, 3905
- stl_iterator_base_funcs.h, 3909
- stl_iterator_base_types.h, 3911
- stl_list.h, 3913
- stl_map.h, 3915
- stl_multimap.h, 3917
- stl_multiset.h, 3919
- stl_numeric.h, 3921
- stl_pair.h, 3923
- stl_queue.h, 3925
- stl_raw_storage_iter.h, 3927
- stl_relops.h, 3928
- stl_set.h, 3929
- stl_stack.h, 3931
- stl_tempbuf.h, 3933
- stl_tree.h, 3934
- stl_uninitialized.h, 3936
- stl_vector.h, 3938
- stossc
 - __gnu_cxx::enc_filebuf, 905
 - __gnu_cxx::stdio_filebuf, 988
 - __gnu_cxx::stdio_sync_filebuf, 1016
 - std::basic_filebuf, 1665
 - std::basic_streambuf, 2187
 - std::basic_stringbuf, 2271
- str
 - std::basic_istringstream, 2003
 - std::basic_ostringstream, 2144, 2145
 - std::basic_stringbuf, 2271
 - std::basic_stringstream, 2334
 - std::match_results, 2908
 - std::sub_match, 3282
- stream_iterator.h, 3940
- streambuf, 3941
 - io, 61
- streambuf.tcc, 3942
- streambuf_iterator.h, 3943
- streambuf_type
 - std::istreambuf_iterator, 2804
 - std::ostreambuf_iterator, 3116
- streamoff
 - std, 598
- streampos
 - std, 599
- streamsize
 - std, 599
- stride
 - numeric_arrays, 114
- string, 3945, 3946
- string_type
 - std::collate, 2434
 - std::collate_byname, 2441
 - std::messages, 2920
 - std::messages_byname, 2925
 - std::money_get, 2934
 - std::money_put, 2940
 - std::moneypunct, 2946
 - std::moneypunct_byname, 2959
 - std::numpunct, 3095
 - std::numpunct_byname, 3103
- stringbuf
 - io, 61
- stringfwd.h, 3949
- stringstream
 - io, 61
- substr
 - __gnu_cxx::__versa_string, 854
 - __gnu_debug::basic_string, 1122
 - std::basic_string, 2250
- subtractive_rng
 - __gnu_cxx::subtractive_rng, 1023
- suffix
 - std::match_results, 2909
- sum
 - numeric_arrays, 114
- sungetc
 - __gnu_cxx::enc_filebuf, 905
 - __gnu_cxx::stdio_filebuf, 988
 - __gnu_cxx::stdio_sync_filebuf, 1016
 - std::basic_filebuf, 1665
 - std::basic_streambuf, 2187
 - std::basic_stringbuf, 2271
- swap

- __gnu_cxx, 368
- __gnu_cxx::__versa_string, 855
- __gnu_debug::basic_string, 1123
- mutating_algorithms, 190
- std, 679–681
- std::basic_regex, 2166
- std::basic_string, 2251
- std::deque, 2569
- std::forward_list, 2632
- std::function< _Res(_- ArgTypes...)>, 2647
- std::list, 2847
- std::map, 2896
- std::match_results, 2909
- std::multimap, 2987
- std::multiset, 3006
- std::set, 3243
- std::vector, 3388
- tr1_regex, 159
- swap_ranges
 - mutating_algorithms, 190
- sync
 - __gnu_cxx::enc_filebuf, 905
 - __gnu_cxx::stdio_filebuf, 988
 - __gnu_cxx::stdio_sync_filebuf, 1016
 - std::basic_filebuf, 1665
 - std::basic_fstream, 1732
 - std::basic_ifstream, 1790
 - std::basic_iostream, 1889
 - std::basic_istream, 1945
 - std::basic_istreamstream, 2003
 - std::basic_streambuf, 2187
 - std::basic_stringbuf, 2272
 - std::basic_stringstream, 2335
- sync_with_stdio
 - std::basic_fstream, 1733
 - std::basic_ifstream, 1790
 - std::basic_ios, 1823
 - std::basic_iostream, 1890
 - std::basic_istream, 1946
 - std::basic_istreamstream, 2004
 - std::basic_ofstream, 2051
 - std::basic_ostream, 2097
 - std::basic_ostreamstream, 2145
 - std::basic_stringstream, 2335
- std::ios_base, 2746
- syntax_option_type
 - std::regex_constants, 745
- system_error, 3950
- t
 - std::binomial_distribution, 2364
- table
 - std::ctype< char >, 2483
 - std::ctype_byname< char >, 2528
- table_size
 - std::ctype< char >, 2486
 - std::ctype_byname< char >, 2531
- tag_and_trait.hpp, 3952
- tags.h, 3955
- tan
 - complex_numbers, 52
- tanh
 - complex_numbers, 52
- target
 - std::function< _Res(_- ArgTypes...)>, 2647
- target_type
 - std::function< _Res(_- ArgTypes...)>, 2648
- tellg
 - std::basic_fstream, 1733
 - std::basic_ifstream, 1791
 - std::basic_iostream, 1890
 - std::basic_istream, 1946
 - std::basic_istreamstream, 2004
 - std::basic_stringstream, 2335
- tellp
 - std::basic_fstream, 1734
 - std::basic_iostream, 1891
 - std::basic_ofstream, 2052
 - std::basic_ostream, 2097
 - std::basic_ostreamstream, 2145
 - std::basic_stringstream, 2336
- temporary_buffer
 - __gnu_cxx::temporary_buffer, 1025
- terminate
 - exceptions, 36
- terminate_handler
 - exceptions, 34
- test

- std::bitset, 2378
- tgmath.h, 3957
- thousands_sep
 - std::moneypunct, 2955
 - std::moneypunct_byname, 2966
 - std::numpunct, 3099
 - std::numpunct_byname, 3106
- thread, 3958
- Threads, 75
- throw_allocator.h, 3960
- throw_with_nested
 - exceptions, 36
- tie
 - std::basic_fstream, 1734, 1735
 - std::basic_ifstream, 1791, 1792
 - std::basic_ios, 1824
 - std::basic_ostream, 1891, 1892
 - std::basic_istream, 1946, 1947
 - std::basic_istream, 2005
 - std::basic_ofstream, 2052, 2053
 - std::basic_ostream, 2097, 2098
 - std::basic_ostringstream, 2146
 - std::basic_stringstream, 2336, 2337
- Time, 37
- time
 - std::locale, 2857
- time_get
 - std::time_get, 3293
- time_members.h, 3963
- time_point_cast
 - std::chrono, 731
- time_put
 - std::time_put, 3314
- tinyness_before
 - std::__numeric_limits_base, 1489
 - std::numeric_limits, 3055
- TLB_size
 - __gnu_parallel::_Settings, 1269
- to_string
 - std::bitset, 2379
- to_ulong
 - std::bitset, 2379
- tolower
 - std, 682
 - std::__ctype_abstract_base, 1403, 1404
- std::ctype, 2470
- std::ctype< char >, 2484
- std::ctype< wchar_t >, 2500
- std::ctype_byname, 2515
- std::ctype_byname< char >, 2528
- top
 - std::priority_queue, 3156
 - std::stack, 3272
- toupper
 - std, 682
 - std::__ctype_abstract_base, 1404
 - std::ctype, 2471
 - std::ctype< char >, 2484, 2485
 - std::ctype< wchar_t >, 2500, 2501
 - std::ctype_byname, 2516
 - std::ctype_byname< char >, 2529
- tr1_math_spec_func
 - assoc_laguerre, 119
 - assoc_legendre, 119
 - beta, 119
 - comp_ellint_1, 119
 - comp_ellint_2, 119
 - comp_ellint_3, 119
 - conf_hyperg, 120
 - cyl_bessel_i, 120
 - cyl_bessel_j, 120
 - cyl_bessel_k, 120
 - cyl_neumann, 120
 - ellint_1, 121
 - ellint_2, 121
 - ellint_3, 121
 - expint, 121
 - hermite, 121
 - hyperg, 121
 - laguerre, 122
 - legendre, 122
 - riemann_zeta, 122
 - sph_bessel, 122
 - sph_legendre, 122
 - sph_neumann, 122
- tr1_regex
 - cregex_token_iterator, 130
 - csub_match, 130
 - isctype, 132
 - operator<, 135–137
 - operator<<, 138

- operator<=, 138–140
- operator>, 144–146
- operator>=, 147–149
- operator==, 141–144
- regex, 130
- regex_match, 149–153
- regex_replace, 153, 154
- regex_search, 155–158
- sregex_token_iterator, 130
- ssub_match, 131
- swap, 159
- value, 160
- wcregex_token_iterator, 131
- wcsub_match, 131
- wregex, 131
- wsregex_token_iterator, 131
- wssub_match, 131
- traits_type
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 969
 - __gnu_cxx::stdio_sync_filebuf, 1002
 - std::basic_filebuf, 1648
 - std::basic_fstream, 1687
 - std::basic_ifstream, 1756
 - std::basic_ios, 1810
 - std::basic_iostream, 1845
 - std::basic_istream, 1913
 - std::basic_istream::sentry, 1957
 - std::basic_istreamstream, 1970
 - std::basic_ofstream, 2025
 - std::basic_ostream, 2072
 - std::basic_ostringstream, 2119
 - std::basic_streambuf, 2173
 - std::basic_stringbuf, 2256
 - std::basic_stringstream, 2290
 - std::istreambuf_iterator, 2804
 - std::ostream_iterator, 3111
 - std::ostreambuf_iterator, 3116
- transform
 - mutating_algorithms, 191
 - std::collate, 2437
 - std::collate_byname, 2443
 - std::regex_traits, 3204
- transform_minimal_n
 - __gnu_parallel::_Settings, 1269
- transform_primary
 - std::regex_traits, 3205
- translate
 - std::regex_traits, 3206
- translate_nocase
 - std::regex_traits, 3206
- traps
 - std::__numeric_limits_base, 1489
 - std::numeric_limits, 3056
- tree_policy.hpp, 3964
- tree_trace_base.hpp, 3965
- trie_policy.hpp, 3966
- true_type
 - metaprogramming, 166
- trunename
 - std::numpunct, 3100
 - std::numpunct_byname, 3106
- trunc
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1832
 - std::basic_iostream, 1901
 - std::basic_istream, 1956
 - std::basic_istreamstream, 2014
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
- try_lock
 - mutexes, 70
- tuple, 3967
- Type Traits, 161
- type_traits, 3970, 3973
- type_traits.h, 3977
- type_utils.hpp, 3978
- typeinfo, 3979
- typelist.h, 3980
- types.h, 3982
- types_traits.hpp, 3984
- u16streampos
 - std, 599
- u32streampos
 - std, 599
- uflow

- __gnu_cxx::enc_filebuf, 906
 - __gnu_cxx::stdio_filebuf, 989
 - __gnu_cxx::stdio_sync_filebuf, 1017
 - std::basic_filebuf, 1666
 - std::basic_streambuf, 2188
 - std::basic_stringbuf, 2272
- uncaught_exception
 - exceptions, 36
- underflow
 - __gnu_cxx::enc_filebuf, 906
 - __gnu_cxx::stdio_filebuf, 989
 - __gnu_cxx::stdio_sync_filebuf, 1017
 - std::basic_filebuf, 1666
 - std::basic_streambuf, 2188
 - std::basic_stringbuf, 2273
- unexpected
 - exceptions, 36
- unexpected_handler
 - exceptions, 34
- unget
 - std::basic_fstream, 1735
 - std::basic_ifstream, 1792
 - std::basic_iostream, 1892
 - std::basic_istream, 1947
 - std::basic_istreamstream, 2006
 - std::basic_stringstream, 2337
- Uniform Distributions, 292
- uniform_int_distribution
 - std::uniform_int_distribution, 3341
- uniform_real_distribution
 - std::uniform_real_distribution, 3346
- uninitialized_copy
 - std, 682
- uninitialized_copy_n
 - SGIextensions, 24
 - std, 682
- uninitialized_fill
 - std, 683
- uninitialized_fill_n
 - std, 683
- unique
 - mutating_algorithms, 192
 - std::forward_list, 2632, 2633
 - std::list, 2847, 2848
 - unique_copy
 - mutating_algorithms, 193
 - unique_copy.h, 3985
 - unique_copy_minimal_n
 - __gnu_parallel::_Settings, 1269
 - unique_ptr.h, 3986
- unitbuf
 - std, 684
 - std::basic_fstream, 1744
 - std::basic_ifstream, 1800
 - std::basic_ios, 1832
 - std::basic_iostream, 1901
 - std::basic_istream, 1956
 - std::basic_istreamstream, 2014
 - std::basic_ofstream, 2061
 - std::basic_ostream, 2106
 - std::basic_ostreamstream, 2155
 - std::basic_stringstream, 2346
 - std::ios_base, 2753
- Unordered Associative Containers, 29
- unordered_map, 3988–3990
- unordered_map.h, 3991
- unordered_set, 3992–3994
- unordered_set.h, 3995
- unsetf
 - std::basic_fstream, 1736
 - std::basic_ifstream, 1792
 - std::basic_ios, 1824
 - std::basic_iostream, 1892
 - std::basic_istream, 1948
 - std::basic_istreamstream, 2006
 - std::basic_ofstream, 2053
 - std::basic_ostream, 2098
 - std::basic_ostreamstream, 2147
 - std::basic_stringstream, 2338
 - std::ios_base, 2746
- unshift
 - std::__codecvt_abstract_base, 1390
 - std::codecvt, 2405
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2411
 - std::codecvt< char, char, mbstate_t >, 2417
 - std::codecvt< wchar_t, char, mbstate_t >, 2423
 - std::codecvt_byname, 2430

- upper_bound
 - binary_search_algorithms, 244, 245
 - std::map, 2896
 - std::multimap, 2987
 - std::multiset, 3006, 3007
 - std::set, 3243, 3244
 - uppercase
 - std, 684
 - std::basic_fstream, 1745
 - std::basic_ifstream, 1801
 - std::basic_ios, 1832
 - std::basic_istream, 1901
 - std::basic_istream, 1956
 - std::basic_istream, 2014
 - std::basic_ofstream, 2062
 - std::basic_ostream, 2107
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2347
 - std::ios_base, 2753
 - use_facet
 - std, 684
 - std::locale::id, 2861
 - Utilities, 76
 - utility, 3996, 3997
 - valarray, 3998
 - numeric_arrays, 90, 91
 - std::valarray, 3366
 - valarray_after.h, 4004
 - valarray_array.h, 4019
 - valarray_array.tcc, 4030
 - valarray_before.h, 4032
 - value
 - tr1_regex, 160
 - value_comp
 - std::map, 2897
 - std::multimap, 2988
 - std::multiset, 3007
 - std::set, 3244
 - value_compare
 - std::set, 3231
 - value_type
 - std::back_insert_iterator, 1635
 - std::complex, 2446
 - std::front_insert_iterator, 2639
 - std::insert_iterator, 2731
 - std::istream_iterator, 2800
 - std::istreambuf_iterator, 2804
 - std::iterator, 2808
 - std::ostream_iterator, 3111
 - std::ostreambuf_iterator, 3116
 - std::raw_storage_iterator, 3182
 - std::reverse_iterator, 3216
 - std::set, 3231
 - vector, 4033, 4034, 4036
 - std::__debug::vector, 1471
 - std::vector, 3371, 3372
 - vector.tcc, 4038
 - vstring.h, 4039
 - vstring.tcc, 4044
 - vstring_fwd.h, 4046
 - vstring_util.h, 4047
 - wcerr
 - std, 686
 - wcin
 - std, 686
 - wclog
 - std, 686
 - wcout
 - std, 686
 - wregex_token_iterator
 - tr1_regex, 131
 - wcsub_match
 - tr1_regex, 131
 - wfilebuf
 - io, 62
 - wfstream
 - io, 62
 - what
 - __gnu_cxx::forced_error, 917
 - __gnu_cxx::recursive_init_error, 950
 - std::bad_alloc, 1637
 - std::bad_cast, 1638
 - std::bad_exception, 1639
 - std::bad_function_call, 1640
 - std::bad_typeid, 1641
 - std::bad_weak_ptr, 1642
 - std::domain_error, 2585
 - std::exception, 2596
 - std::future_error, 2658
-

- std::invalid_argument, 2734
- std::ios_base::failure, 2755
- std::length_error, 2812
- std::logic_error, 2864
- std::out_of_range, 3119
- std::overflow_error, 3122
- std::range_error, 3167
- std::regex_error, 3189
- std::runtime_error, 3223
- std::system_error, 3285
- std::underflow_error, 3338
- widen
 - std::__ctype_abstract_base, 1405
 - std::basic_fstream, 1736
 - std::basic_ifstream, 1793
 - std::basic_ios, 1825
 - std::basic_iostream, 1893
 - std::basic_istream, 1948
 - std::basic_istream, 2006
 - std::basic_ofstream, 2053
 - std::basic_ostream, 2098
 - std::basic_ostringstream, 2147
 - std::basic_stringstream, 2338
 - std::ctype, 2472
 - std::ctype< char >, 2485, 2486
 - std::ctype< wchar_t >, 2501, 2502
 - std::ctype_byname, 2517
 - std::ctype_byname< char >, 2530
- width
 - std::basic_fstream, 1736, 1737
 - std::basic_ifstream, 1793, 1794
 - std::basic_ios, 1825
 - std::basic_iostream, 1893, 1894
 - std::basic_istream, 1949
 - std::basic_istream, 2007
 - std::basic_ofstream, 2054
 - std::basic_ostream, 2099
 - std::basic_ostringstream, 2147, 2148
 - std::basic_stringstream, 2338, 2339
 - std::ios_base, 2746, 2747
- wfstream
 - io, 62
- wios
 - io, 62
- wiostream
 - io, 62
- wistream
 - io, 62
- wistringstream
 - io, 62
- wofstream
 - io, 63
- workstealing.h, 4048
- wostream
 - io, 63
- wostringstream
 - io, 63
- wregex
 - tr1_regex, 131
- write
 - std::basic_fstream, 1737
 - std::basic_iostream, 1894
 - std::basic_ofstream, 2054
 - std::basic_ostream, 2100
 - std::basic_ostringstream, 2148
 - std::basic_stringstream, 2339
- ws
 - std, 685
- wsregex_token_iterator
 - tr1_regex, 131
- wssub_match
 - tr1_regex, 131
- wstreambuf
 - io, 63
- wstreampos
 - std, 599
- wstringbuf
 - io, 63
- wstringstream
 - io, 63
- xalloc
 - std::basic_fstream, 1738
 - std::basic_ifstream, 1794
 - std::basic_ios, 1826
 - std::basic_iostream, 1895
 - std::basic_istream, 1949
 - std::basic_istream, 2008
 - std::basic_ofstream, 2055
 - std::basic_ostream, 2100
 - std::basic_ostringstream, 2149
 - std::basic_stringstream, 2340

std::ios_base, [2747](#)

xsggetn

- [__gnu_cxx::enc_filebuf](#), [907](#)
- [__gnu_cxx::stdio_filebuf](#), [990](#)
- [__gnu_cxx::stdio_sync_filebuf](#),
[1018](#)
- [std::basic_filebuf](#), [1667](#)
- [std::basic_streambuf](#), [2189](#)
- [std::basic_stringbuf](#), [2273](#)

xspn

- [__gnu_cxx::enc_filebuf](#), [907](#)
- [__gnu_cxx::stdio_filebuf](#), [991](#)
- [__gnu_cxx::stdio_sync_filebuf](#),
[1018](#)
- [std::basic_filebuf](#), [1668](#)
- [std::basic_streambuf](#), [2190](#)
- [std::basic_stringbuf](#), [2274](#)

yield

- [std::this_thread](#), [757](#)