This document lists the resources needed for the installation and explains the steps to install the Scorpion System on the following platforms.

Platforms Supported	
Machine	Operating System
DEC VAX	4.3BSD UNIX
DEC 3100	V2.0 (rev. 7)ULTRIX
HP 9000	HP-UX
NeXT	Mach 1.0
Sun-3, -4	SunOS Release 3.3 or 4.0
Sequent Symmetry	Dynix 3.0

We strongly suggest that you read the entire document before proceeding with the installation.

1. Required Resources

There are two ways to acquire the Scorpion Release 5.0. You may receive it by FTP (File Transfer Protocol) or on a tape. The tape is recorded at 1600 bpi. A tarred and compressed file, `scorpion5.0.tar.Z`, contains the entire system and occupies about 6MB.

1.1. Disk Space

The required object code, libraries, and include files for using the system will occupy approximately 9 Mbytes of disk space after building; the required and optional files together occupy a total of 37 Mbytes. To build the system will temporarily require an additional approximately 21–30 Mbytes of disk space, over that needed for the destination directories.

The following table lists the approximate disk space requirements for each of the steps (steps that are required are indicated as such.) The first three (required) steps, `make InstallInclude`, `make InstallHelp`, and `make InstallMisc` together require about 0.2MB of disk space, and are not shown in the table. These values are approximate; mileage may vary depending on the target architecture and the details of the configuration selected.

Disk Space Requirements (in MBytes)		
Step	MBytes	Required?
Original Installation directory	18.0	Required
`Libraries`	4.8	Required
`InstallLibraries`	4.3	Required
`Executables`	16.0	Required
`InstallExecutables`	4.6	Required
`Clean`	-8.6	Optional
`InstallSrc`	22.2	Optional
`InstallMan`	0.4	Optional
`InstallDoc`	1.5	Optional
`InstallExamples`	0.8	Optional
`Checkout`	2.2	Optional
`CompileExamples`	4.0	Optional

These figures assume that the system has been compiled in optimize mode (see below). If the system is compiled in debug mode, the size of the executables and the library will increase substantially, perhaps by a factor of 2 or 3.

1.2. Main Memory and Swap Space

The installation should be possible in 4MB of main memory, though 8MB is certainly recommended. 16MB of swap space and about 2MB of tmp space should be sufficient for the installation. If the swap space is exceeded, the error message ‘‘Ran out of heap storage’’ will appear. The checkout (`make Checkout`) as well as runs of `idlc` on large inputs may require more main memory or swap space. All of our development has been on machines with at least 8MB of main memory and 24MB of swap space.

1.3. Execution Time

The installation takes between 2 and 7 hours, depending primarily on the type of machine and file server. The break-down by steps is shown in the following table.

All times are in minutes, and assume an otherwise unloaded machine and lightly loaded file server. The first three (required) steps, `make InstallInclude`, `make InstallHelp`, and `make InstallMisc` together took about 1 minute, and are not included in the table.

Approximate Execution Time (in minutes)								
Step	Required?	Sun3	Sun4	Vax	Sequent	DEC3100	HP9000	NeXT
`Libraries`	Required	13	4	9	8	10	30	7
`InstallLibraries`	Required	1	1	1	1	1	1	1
`Executables`	Required	180	59	85	93	83	260	94
`InstallExecutables`	Required	3	2	2	1	4	5	3
`Clean`	Optional	2	1	1	1	1	2	1
`InstallSrc`	Optional	3	2	5	3	7	9	4
`InstallMan`	Optional	1	1	1	1	1	1	1
`InstallDoc`	Optional	1	1	1	1	2	1	1
`InstallExamples`	Optional	1	1	1	1	2	1	1
`Checkout`	Optional	6	3	8	7	6	34	6
`CompileExamples`	Optional	39	7	15	18	11	30	11
Total	Both	250	82	129	135	126	376	130

2. Machine Dependencies

The following instructions explain the steps for installing the Scorpion System on the platforms previously listed. These instructions vary depending on the machine and operating system on which the Scorpion System is installed. Specifically, the following potential problems are addressed in the installation procedure.

- There are bugs in `yacc` on the DEC3100, Sun-3 and Sun-4; it either core dumps or produces bad code. Therefore, we include the generated C files in the distribution and skip running `yacc` on these machines.

- A C file used in `idlread` and generated by `lex` on a DEC3100 and Sequent Symmetry sometimes core dumps when executed. Therefore, we include the generated C file in the distribution and skip running `lex` on the DEC3100.

- `cc` loops indefinitely on Sun-4’s when run on large files (> 6,000 lines) with the optimization flag set. We therefore do not compile large C files in `idlc` with the compilation flag set.

- The `signal` routine is inconsistently declared in various versions of Unix. We have attempted to use the correct version for all machines via conditional compilation. The files `scorpion/src/IDLlib/libci/runv.c` and `runvp.c` may need to be changed if an error occurs during installation.

- On HP9000 systems `cc` fails on large C files with a symbol table overflow message. We suggest including the `+Ns5000` option in the `MYCFLAGS` variable.

  Two other machine differences are reflected in the `Makefile`s.

- `pc` on the Suns runs `cpp` initially on the source Pascal file, allowing macros to be used. There are two version of the billing example provided, one using macros and one not using them. `make CompileExamples` chooses the appropriate version.
- `cpp` on the different machines varies in its handling of white space, which causes both `gentransitions` and `xref` to produce slightly different results. Hence `make Checkout` uses different result files for the different machine types.

3. Installation Instructions

To install the Scorpion System, perform the following steps in order. Only the first step depends on whether you acquired the system by tape or via FTP. While there are a substantial number of definitions to consider, in most cases the default value will suffice. If this system is being installed on multiple architectures, then steps 5 and onward should be repeated for each architecture. Note that `idlview_master` should be installed on only one machine, and thus one architecture, in a local area network. A variable discussed below controls whether or not this program should be installed.

We suggest that you read through this entire document before performing any of the steps, to anticipate problems before they occur.

Makefile variables must have the no spaces between the '=' and the name of the variable. For example,

```
HASPASCAL= false
```

is okay and so is

```
HASPASCAL=true
```

but

```
HASPASCAL =true
```

causes problems when updating lower level makefiles. Also, do not put any comments on the same line as the variable definition. The following will cause problems.

```
TOPLEVELDESTDIR=/usr/scorpion # was /usr/local/scorpion
```

Finally, you should directly modify the `Makefile` in the `scorpion` directory, rather than invoking it with Makefile variables specified.

(1) Change your current working directory to a directory where you want the contents of the Scorpion System to be stored until the installation is complete. The executables will be made in this directory, then moved over to a separate *destination* directory as a separate step.

`cd` *somedir*

From Tape: Mount the tape on your system's tape drive making sure it is not write-enabled. The tape is read by typing

```
tar x
```

In most configurations, this will default to reading at 1600bpi, which is the density at which the tape was written. A file named `scorpion5.0.tar.Z` will be created in *somedir*.

Via FTP: Copy the file `scorpion5.0.tar.Z` into *somedir* from the FTP area.

(2) Uncompress the file by running:

```
uncompress scorpion5.0.tar.Z
```

The file `scorpion5.0.tar` will be created when `uncompress` successfully terminates.

(3) The Scorpion System is extracted by untarring the file created above. This may be done as follows and will take several minutes.

```
tar xf scorpion5.0.tar
```

(4) A directory `scorpion` will be created in *somedir*. Print the file *somedir*`/scorpion/NOTES`; this discusses important details on the release not covered by the installation instructions. Also print the file *somedir*`/scorpion/README`.

(5) To reduce disk space the old tar file may now be deleted:

```
rm scorpion5.0.tar
```

Change to this new directory:

```
cd scorpion
```

(6) This directory contains a file named `Makefile`. The system will be stored by default in the destination directories listed below. To change this default, edit the definitions in `Makefile`. These directories must already exist. If all the destination directories are to reside under the same directory, simply indicate by setting `TOPLEVELDESTDIR`. The default value of this variable is:

```
TOPLEVELDESTDIR=/usr/local/scorpion
```

All the destination directories may be rooted under one directory (e.g., `/usr/local/scorpion`) as above, or may be in different directories. This installation procedure provides for both options. Just set the variables accordingly. If any of the destination directories below is not rooted under `TOPLEVELDESTDIR`, it must be separately specified. **Please set these variables very carefully, as they are used extensively. None of the destination directories may be in** ***somedir*****`/scorpion` or the installation will fail.**

```
INCDEST=$(TOPLEVELDESTDIR)/include
BINDEST=$(TOPLEVELDESTDIR)/bin
LIBDEST=$(TOPLEVELDESTDIR)/lib
SRCDEST=$(TOPLEVELDESTDIR)/src
MANDEST=$(TOPLEVELDESTDIR)/man
HELPDEST=$(TOPLEVELDESTDIR)/help
MISCDEST=$(TOPLEVELDESTDIR)/misc
DOCDEST=$(TOPLEVELDESTDIR)/doc
EXAMPLEDEST=$(DOCDEST)/examples
```

Make sure there are no programs with identical names as those of the system (see the man pages provided in a separate document for the names of the programs in the system).

Errors may occur during installation if any of the `DEST` directories are in *somedir*`/scorpion` and if installation is requested into those directories. The installation must be run by a user with permission to write in all of the destination directories.

(7) Indicate the machine type on which Scorpion is to be installed by setting the `MACHINE` macro in the `Makefile` to either `sun` (for a sun-3 machine), `sparc` (for a sun-4 machine) `vax`, `mips`, `hp9000`, `next`, or `sequent`. The default is `sparc`. See the discussion in Section 2 above on how this macro is used.

```
MACHINE=sparc
```

(8) Set the following to either `true` or `false` in the `Makefile`. `RUNYACC` should be set to `false` if `yacc` must not be run during the installation. This may be necessary for some machine types on which `yacc` does not work well. A similar reasoning applies to `RUNLEX` if `lex` has problems on your machine for large input files. **We strongly recommend setting both** `RUNYACC` **and** `RUNLEX` **to false.** `LINTLIBRARY` should be set to `false` if `lint` will not be run on the system, otherwise setting it to `true` will create lint libraries corresponding to the libraries in the system. Setting `DEBUGLIBRARY` to `true` produces additional libraries which contain information necessary for debugging. If Pascal is not available on your system, set the `HASPASCAL` flag to `false`. They initially have the indicated values.

```
RUNYACC=false
RUNLEX=false
LINTLIBRARY=true
DEBUGLIBRARY=true
HASPASCAL=true
```

(9) The entire system may be built in `OPTIMIZE` or `DEBUG` mode. The following variable may be modified as needed. In `DEBUG` mode additional symbol table information is stored for each tool in the system. This requires extra disk space and main memory, but allows the tools to be debugged later.

```
HOWTOCOMPILE=OPTIMIZE
```

(10) The following flags are needed to compile C and Pascal files. Don't insert any optimization or debugging related flags in `MYCFLAGS` or `MYPFLAGS`. Optimization related flags should be assigned to `OPTIMIZE_CFLAGS` or `OPTIMIZE_PFLAGS`. Similarly, debugging related flags should be placed in the `DEBUG_CFLAGS` and `DEBUG_PFLAGS` variables. The following settings of the flags are highly recommended. Equivalent settings available on your compiler may be used.

```
MYCFLAGS=-I$(INCDEST)
DEBUG_CFLAGS=-g
OPTIMIZE_CFLAGS=-O

MYPFLAGS=-w
DEBUG_PFLAGS=-g
OPTIMIZE_PFLAGS=-O
```

As discussed in Section 2, above, we suggest including the `+Ns5000` C flag for the HP9000 in the `MYCFLAGS` variable.

(11) It may also be necessary to specify different paths or options for the UNIX utilities used throughout the makefiles. These can be modified by editing the following lines in `Makefile`. To determine the

appropriate path for each utility, execute `misc/wherearethey`. **Please check the following variables very carefully, because they are used extensively in the installation. Every site will most likely have to change at least some of these variables.**

```
CHMOD=/usr/bin/chmod 755
CHOWN=/etc/chown bin
CHGRP=/usr/bin/chgrp scorpion

AR=/usr/bin/ar ru
CAT=/bin/cat
CC=/bin/cc
CP=/bin/cp
FIND=/usr/bin/find
LEX=/usr/bin/lex
LINT=/usr/bin/lint
MAKE=/bin/make
MV=/bin/mv
PC=/usr/ucb/pc
RANLIB=/usr/bin/ranlib
RM=/bin/rm -f
RMDIR=/usr/bin/rmdir
SED=/usr/bin/sed
SHELL=/bin/sh
STRIP=/usr/bin/strip
TOUCH=/usr/bin/touch -c
YACC=/usr/bin/yacc -d
```

The installation will set the owner of the executable files as `bin`, the group as `scorpion`, and the ownership rights to `755`. If the default `CHOWN` is used, the installation will require super-user privileges. If any of the above is not needed or cannot be used (e.g if you don't have super-user privileges), set it to `$(NOOP)`, a command that does nothing. As an example, if you don't want to change the ownership, edit the `CHOWN` variable to:

```
CHOWN=$(NOOP)
```

If `gcc` is preferred over `cc`, then it should be specified using the `CC` variable. Additionally the `-pedantic` option, if desired, must be specified in the `MYCFLAGS` variable, discussed in a previous step. The Scorpion System, and files generated by the system, are ANSI C compatible. At times `gcc` complains about system include files. For this purpose we have included an sed script `misc/gcc.sed` to purge these warnings from the output.

(12) The following commands are used in document processing. They will be used when you run off the documents included in the system. Some documents are in `TeX` format, some in `LaTeX` format, and some in `troff` format (-me macros); several pages are in Postscript. The printer is assumed to be able to handle Postscript or DVI (`TeX` device independent) files. If your printer formats differ substantially from the default ones, you may need to edit the `Makefile`s in the *docdest* directory.

```
TEX=/usr/local/tex

LATEX=/usr/local/latex
```

```
EQN=/usr/bin/eqn
PIC=/usr/local/pic
SOELIM=/usr/ucb/soelim
TBL=/usr/bin/tbl
PSROFF=/usr/local/psroff

PRINT=/usr/ucb/lpr -Plw0
PRINTDVI=/usr/ucb/lpr -Plw0 -d
```

`psroff` is a `troff`-compatible formatter that generates postscript output.

(13) To install `idlview` a number of variables need to be set. `idlview` is a window-based debugging display program. `idlview` must only be installed if either of the following window systems are available on the target machine: SunView, or X Windows. To include `idlview` in the installation set the following variable to `true`. If it is set to `false`, then the rest of this section may be skipped.

```
INSTALLIDLVIEW=true
```

The name of the window system (SunView, or X Windows) that `idlview` is to be configured for should be specified by setting the following variables to `true` or `false`. You may find references to Open Windows in the `Makefile`, but it is only there for future usage. It is not supported in this release.

```
HASSUNTOOLS=true
HASXWINDOWS=true
```

Multiple window systems may be simultaneously supported. For each of the window systems, the appropriate libraries to be linked must be specified. The defaults are given below. Note that the '`-l`' option has been used. If the libraries are in directories unknown by the linker, then the '`-L`' may be specified with the pathnames of possible library directories in `MYCFLAGS`. Alternatively, the full pathname of the library could be specified.

```
SUNLIB=-lsuntool -lsunwindow -lpixrect
XLIB=-lXaw -lXmu -lXt -lXext -lX11
```

The default window manager is also required. This will be the default window system assumed if none is specified in the command line when invoking `idlview` from the shell. The possible values are `SUNTOOLS` (SunView), or `XWINDOWS` (X Windows.)

```
DEFAULTWM=SUNTOOLS
```

`idlview` may be installed separately on several systems with different architectures (e.g. Sun-3, Sun-4.) To do so, simply execute the following steps, including `installExecutables` for each architecture. However, there need only be one `idlview` master process for each local area network, so the following variable should be set to `true` for only one architecture. `idlview_master` is a server program that coordinates binding `idlview` executions to debugger executions.

```
COMPILEMASTER=true
```

The name of the machine on which the `idlview_master` will run must be specified.

```
MASTERHOST=cheltenham
```

`idlview` communicates with `idlbrowse`, when the `explain` button is pressed. A file is required for the information transfer and should be specified by setting the variable below.

```
IDLBROWSEFILE=/tmp/IDLbrowse.info
```

Finally, the path name of the file used by idlview to store internally used information must be specified by setting the following variable.

```
MASTERFILE=/tmp/IDLidlview_master.status
```

Both of the above files will be quite short.

(14) `lint` may be run on the source files. To set the flags used by `lint` change the variable below. The preset value is the default for the system. When these flags are specified, `lint` should print few error messages.

```
LINTFLAGS=-abx
```

(15) The `idlc` translator will not be run during the installation, because of `make TouchFiles`, but will be run during `make Checkout` and `make CompileExamples`. In the following, we recommend that you not use toggle options (such as `-nc`) that would have exactly the opposite of the desired effect if they also appeared in lower level Makefiles.

```
IDLC=$(BINDEST)/idlc
IDLCFLAGS=-v
```

At this point, the `Makefile` has been customized to your installation. When the installations are conducted, the `Makefile`s of the destination directories will be updated to reflect changes which were made above.

(16) To verify that the path settings of the UNIX tools are correct, type

```
make TestPath
```

If any serious errors occur, recheck the paths before continuing. This step may be repeated as needed.

(17) To ensure that `idlc` is not run during the installation, type

```
make TouchFiles
```

This will touch some of the generated files, to ensure that their modification date is after their precursors, so that idlc (always), and yacc and lex (if specified), will not need to be run during the installation.

(18) Type the following command to process the variable settings made above. It also updates **all** the makefiles in the system including the current `Makefile`. During the installation, these new makefiles will be copied to the destination directories.

```
make UpdateMake
```

A copy of `Makefile` is made in `ORIGINAL.Makefile`. If due to any errors the original `Makefile` is clobbered, it may be copied back. Errors in this step should be addressed before continuing with the installation.

These last two steps need only be done once; the remaining steps may be executed multiple times. However, running these steps more than once should not pose any problems. But if you notice any mistakes in variable settings during the installation especially after some libraries or tools have already been compiled, it is safer to restart at this step and perform the following steps again.

(19) To build the binary executables for the Scorpion System and install the programs type

```
make AllInstall
```

This command will compile all the programs, create the libraries and install the libraries, the include files, the binaries, the source files, the man pages, and the examples. Or, you may perform the following steps if you want to have more control of the process.

`make InstallInclude`	-- *install the include files into* `INCDEST`
`make InstallHelp`	-- *install the help files for* `errordb` *into* `HELPDEST`
`make InstallMisc`	-- *install miscellaneous files needed by* `errordb, idlview` and `dbx` *into* `MISCDEST`
`make Libraries`	-- *build the libraries and lint libraries that Scorpion uses*
`make InstallLibraries`	-- *install the Scorpion error information file, and the libraries into* `LIBDEST`
`make Executables`	-- *build the executable programs*
`make InstallExecutables`	-- *install the directly executable programs into* `BINDEST` *and the nondirectly executable programs into* `LIBDEST`
`make Clean`	-- *Remove all intermediate files created by the above steps*
`make InstallSrc`	-- *install the source files in* `SRCDEST`
`make InstallMan`	-- *install the man pages in* `MANDEST`
`make InstallDoc`	-- *installs the documents in* `DOCDEST`
`make InstallExamples`	-- *install the examples in* `DOCDEST/examples`

The steps `make InstallInclude`, `make InstallHelp`, `make InstallMisc`, `make Libraries` and `make InstallLibraries` must be performed first and in the order shown. However, you may perform `make Executables` followed by `make InstallExecutables` after the source files, man pages or the examples are installed. It is not necessary that you install the source files, the man pages or the examples. Depending on the amount of free disk space you may choose to omit them. See Section 1.1 for a detailed breakdown of disk space requirements.

Throughout the installation, a very few minor error messages will appear, but will always be preceded by a message indicating that the message should be ignored. Any error not preceded by such a message is significant, and should be addressed. If any such errors are encountered, the step containing the error or `make AllInstall` can usually be redone after correcting the error, as many times as necessary.

After the installation is complete, if any programs or libraries need to be recompiled, consult the individual `Makefile` for the proper target to use in the `make` command. Since `make` variables are not passed down in this case, the `Makefile` may need to be altered in minor ways.

(20) If you have all the destination directories rooted under one directory, say *topleveldestdir*, then execute the following command

`cp Makefile` *topleveldestdir*`/Makefile.scorpion`

If the source directory were to be modified, then the system could be reconstructed by issuing a

```
make -f Makefile.scorpion System
```

in the *topleveldestdir* directory. This `make System` will not address the problems listed in Section 2.

(21) To check that the Scorpion System tools are functioning properly, execute the following (but only if the examples have been installed, as one of the checks uses files in the example directory).

`cd` *somedir*`/scorpion`
`make Checkout`

A sequence of commands will be printed for each tool which make up the checkout for that tool. For instance for the tool `candleexpand` the following output will result from a successful checkout (a few carriage returns have been inserted for readability).

```
rm -f test/candleexpandtest.out2
make 'SRCDIR=/usr/local/scorpion/src'
      'SPECSDIR=/usr/local/scorpion/src/specs'
      'BINDIR=/usr/local/scorpion/bin' Checkout1
cd /usr/local/scorpion/src/idlc/test; make -s
      'SPECSDIR=/usr/local/scorpion/src/specs'
      'BINDIR=/usr/local/scorpion/bin' test.Cdl
`test.Cdl' is up to date.
/usr/local/scorpion/bin/candleexpand
      </usr/local/scorpion/src/idlc/test/test.Cdl
      >test/candleexpandtest.out2
sh -c 'cmp -s test/candleexpandtest.out1 test/candleexpandtest.out2
      &&(echo "files candleexpandtest.out1 candleexpandtest.out2 agree";
      echo "candleexpand PassedInstallation Check") ||
      (echo "*** candleexpand Failed Installation Check ***") '
files candleexpandtest.out1 candleexpandtest.out2 agree
candleexpand Passed Installation Check
```

In general the last message of any tools checkout must report that it passed the checkout. On the other hand, if a tool is not functioning properly, the last message for that tool will report failure. For instance if `candleexpand` fails, the following will be output.

```
rm -f test/candleexpandtest.out2
make 'SRCDIR=/usr/local/scorpion/src'
      'SPECSDIR=/usr/local/scorpion/src/specs'
      'BINDIR=/usr/local/scorpion/bin' Checkout1
cd /usr/local/scorpion/src/idlc/test; make -s
      'SPECSDIR=/usr/local/scorpion/src/specs'
      'BINDIR=/usr/local/scorpion/bin' test.Cdl
`test.Cdl' is up to date.
/usr/local/scorpion/bin/candleexpand
      </usr/local/scorpion/src/idlc/test/test.Cdl
      >test/candleexpandtest.out2
sh -c 'cmp -s test/candleexpandtest.out1 test/candleexpandtest.out2
      &&(echo "files candleexpandtest.out1 candleexpandtest.out2 agree";
      echo "candleexpand PassedInstallation Check") ||
      (echo "*** candleexpand Failed Installation Check ***") '
*** candleexpand Failed Installation Check ***
```

If this occurs, review the previous steps to ensure that everything was done correctly. If there are still problems, then look at the files being compared with diff to see if there is indeed a problem. Diff is very picky, and some utilities, such as cpp, operate slightly differently on the various machines, sometimes confusing diff. **Just because `diff` says that a specific program failed an installation check doesn't necessarily mean that the program is operating incorrectly; the program may have operated perfectly well.** If the files that diff is comparing look very similar, then there probably is *not* a problem.

If there are still problems, check with the Scorpion System developers, who can be reached at `scorpion-project@cs.arizona.edu`

(22) If you have installed the examples, you can compile them in the installation directory by typing

```
make CompileExamples
```

This also serves as another fairly comprehensive test of the installation.

(23) If you have installed the man pages, you have to expand the pathnames of the files mentioned in the `FILE` section in these man pages to include the correct prefix. `errordb.1`, `gentransitions.1`, `idlc.1`, `idlread.1`, `idlview.1`, `lister.1`, `treewalk`, `xref`, `xref_xref.1`, `ci.3`, `idlio.3`, `liberr.3`, `libidl.3`, `Candle.5`, and `DianaRev4.5`. Moreover, the pathnames of some of the files mentioned in the documents may no longer be correct. The documents themselves contain a note in this regard. A `whatis` database may be created for the manpages. The following works on a Sun-3, Sun-4, NeXT and DEC VAX. See `catman`(1) for more details.

`catman -M` *mandest* `-w`

(24) All the PostScript and DVI files for the documentation have been included in the system. To print out the entire documentation (approximately 500 pages), type

```
cd docdest
make -f Makefile.scorpion PrintDoc
```

Note that the man pages are *not* printed by this command; they must be printed manually.

If you wish, you can recreate the documentation by typing the following in *docdest*.

```
make -f Makefile.scorpion System
```

If your printer programs differ substantially from the default ones specified above, you may need to edit the `Makefile`s in *docdest*.

(25) *idlview_master* may be placed in the system /etc/rc.local (or equivalent) file, so that it may be run whenever the system is booted. For now, you may wish to start the program once the installation is complete.

```
bindest/idlview_master &
```

(26) If the system is working properly, the temporary directory *somedir*`/scorpion` may be deleted, if needed destination directories are different. In that case, type

```
cd somedir
rm -fr scorpion
```

4. Lint

For `lint` to be executed the source must have been installed. If so, then change directory to the new source directory, say *destdir*. To run `lint` on the entire system, simply type

```
cd srcdest
make -f Makefile.scorpion Lint >& lint.out
```

The `lint` libraries are created during the `make Libraries` phase of the installation if `LINTLIBRARY` is set to true. Thus only after the `make InstallLibraries` phase may `lint` be run. The `lint` flags supported are `-abx`. This is the default and may be changed by editing the `LINTFLAGS` variable. Generated source files from `idlc`, `yacc` and `lex` result in many warnings which may be ignored. A sed script `lint.sed` is provided to remove such warnings from the `lint` output.

```
sed -f miscdest/lint.sed < lint.out > lint.clean
```

Additionally, ignore messages printed by the Makefile may precede a tool's `lint` output. Any other errors should be addressed.

5. Acknowledgements

Richard Snodgrass directed the Scorpion System design and implementation, including all releases, and prepared Release 4.1. Karen Shannon prepared and coordinated the installation instructions and distributions for Release 2.0 and 3.0, and helped direct Release 4.0 onward. Sundar Varadarajan prepared Release 3.2. Edilberto Uichanco prepared internal Release 3.4. Clement Cheung prepared Release 4.0.

Releases 2.0 through 4.1 were done at the University of North Carolina. Hasnain Karampurwala prepared Release 4.2 and Release 5.0 at the University of Arizona.

We also wish to acknowledge support through the years from the following. IBM provided partial support through a Junior Faculty Award to Richard Snodgrass. This work was supported in part by the National Science Foundation through grants DCR-8402339 and IRI-8902707, and through CISE IIP grants CDA-8722752 (to the Department of Computer Science, the University of North Carolina) and CDA-8822652 (to the Department of Computer Science, the University of Arizona), and by the Office of Naval Research under contract N00014-86-K-0680 to the Department of Computer Science, UNC. The Software Engineering Institute at Carnegie Mellon University indirectly provided support.