Porting Open Source Software to SCO

Dr. Ronald Joe Record
Open Source Program Architect
SCO

Much of the most useful and popular software today is published and distributed in source form. The Apache web server, KDE, the Mtools DOS utilities and the GNU C compiler are a few examples of what is commonly referred to as Open Source software. This article attempts to provide some insight, tips, techniques and methodology for building Open Source software on and for SCO OpenServer 5, UnixWare 7 and in general.

Setting Up Your Development Environment

Perhaps the single most important step in porting *any* software is the creation of an appropriate build environment. Fortunately, on SCO OpenServer 5 and UnixWare 7, much of the work has already been done for you. On either platform, simply install the base Operating System, native development system, Java Development Kit and the SCO Skunkware CD included in the Operating System media kit.

If you are not a licensed Development System user you can still build an appropriate development environment by utilizing the GNU development system along with the native libraries and headers. On OpenServer you will need to install the "Linkers and Application Libraries" package of the Development System (no license key is required). On either platform you can simply install the Development System sans license key.

After having installed the Operating System, development system of choice, and SCO Skunkware you will now have access to the standard tools necessary to build most Open Source packages. These tools include the GNU Compiler Collection, Bison, Flex, GNU Make, autoconf, automake, and a wide variety of header files and libraries.

Where and How to Obtain Open Source Software

There are a number of excellent Open Source "portals" on the Internet. These are sites which provide links and references to the thousands of Open Source projects. Good starting points include:

- The SCO Skunkware web site at http://www.sco.com/skunkware. Each of the Skunkware components described here contains a link to the binary distribution for either OpenServer or UnixWare 7 as well as a link to the original source and project home page (if any). Additionally, there's a table of links at http://www.sco.com/skunkware/links.html.
- The Freshmeat web site at http://freshmeat.net.
- The FileWatcher web site at http://filewatcher.org.
- The WebWatcher web site at http://webwatcher.org.
- Sunsite's Linux archive at ftp://sunsite.unc.edu/pub/Linux/welcome.html. Note that many sites indicate their Open Source projects are for Linux when, in fact, the source is portable across most modern UNIX platforms as well.
- The Open Source Who's Who http://support.lcg.org/Whoswho/.
- The Free Software Foundation FTP download area at ftp://ftp.gnu.org/gnu/.

Another method for obtaining access to the very latest development sources for many Open Source projects is Anonymous CVS. After installing SCO Skunkware you will have the necessary CVS utilities in `/usr/local/bin`. To download a project's source tree via CVS, login to the CVS server with the project password, checkout the desired source, and logout. The project home page will supply you with the CVS server name and password if they are maintaining an anonymous CVS source repository. I've written a script to automate this procedure for several projects. You can download this script via http://www.sco.com/skunkware/src/devtools/cvsget.gz

Configuring Open Source Software for Compilation

After identifying an Open Source package of interest, locating it, and downloading the source archive it is typically the case that the source needs to be unpacked and configured for your platform. Most source archives downloadable over the Internet are compressed tar archives. To extract on of these, say `widget-1.0.tar.gz`, issue the command:

```
gzcat widget-1.0.tar.gz | tar xf -
```

It is often prudent to first simply list the contents of the archive to determine if it extracts into it's own subdirectory. If not, create a subdirectory and extract the archive from there.

Once extracted the source project needs to be configured for your platform. Most modern Open Source projects use GNU Autoconf to provide configuration scripts which attempt to automate the configuration process. In the simplest cases it will only be necessary to run the `configure` script:

```
cd widget-1.0 ; ./configure
```

Unfortunately, it is often necessary to "help" `configure` understand your platform. For instance, on UnixWare 7 it is sometimes necessary to:

- replace configure's "-lnsl" with "-lsocket -lnsl"
- replace configure's "-lICE" with "-lSM -lICE"
- order "-lSM -lICE" after "-lXt" but prior to "-lX11" on the link line
- rearrange the order of certain graphics libraries like JPEG, TIFF, PNG or others

A typical link line for a graphical application might include:

```
-L/usr/local/lib -lXpm -ltiff -ljpeg -lpng -lz -lXaw -lXmu -
lXt -lSM -lICE -lXext -lX11 -lm -lsocket -lnsl -lgen
```

Many of these idiosyncracies, due in part to the single-pass nature of the native linker, can be simply accomodated by rerunning autoconf which regenerates the `configure` script from the `configure.in` file for your platform.

Another configuration area that often causes problems is the building of shared libraries. The `ltconfig` script contains the platform-specific instructions and options for creating shared libraries. In order to build shared libraries on UnixWare 7 it is necessary to modify `ltconfig` and add the following to the appropriate case statements:

```
sysv5*)
  pic_flag='-KPIC'
  link_static_flag='-Bstatic'
  wl='-Wl,'
  ;;

sysv5*)
  archive_cmds='$LD -G -h $rpath/$soname -o $lib$libobjs'
  runpath_var=LD_RUN_PATH
  hardcode_runpath_var=yes
  hardcode_shlibpath_var=no
  ;;

sysv5*)
  version_type=osf
  soname_spec='$libname.so.$major'
  library_names_spec='$libname.so.$versuffix $lib-
name.so.$major $libname.so'
  shlibpath_var=LD_LIBRARY_PATH
  ;;
```

By default, `configure` will use the GNU C Compiler. If you wish to use an alternate compiler the CC environment variable can be set appropriately. On UnixWare 7 the Skunkware components are built with the native compiler for performance and portability reasons. As a convenience i've constructed simple shell scripts to front-end the `configure` script. For instance, on UnixWare 7 to use the native C compiler:

```
#!/bin/sh
HOST=i586-sco-sysv5
[ -f mout-config ] && mv mout-config mout-config$$
```

```
CC="cc"
CPP="$CC -E"
CFLAGS="-Xa -Dasm=__asm -DANSICPP -O3 -Kthread -Kalloca -
I/usr/local/include -L/usr/local/lib"
CXX="CC"
CXXFLAGS="-O3 -I/usr/local/include/stl -I/usr/local/include -
L/usr/local/lib"
RANLIB=true
MAKE=/usr/local/bin/make
export CC CPP CXX RANLIB MAKE CFLAGS CXXFLAGS
./configure $* $HOST 2>&1 | tee mout-config
```

Note the use of "-Kthread -Kalloca". These enable threads and the built-in alloca function. The check for threads support in `configure` often fails on UnixWare 7. If you wish to include threads support it may be necessary to modify `configure` and/or `config.h`. On SCO OpenServer the use of threads can be accomplished via the FSU Pthreads package on Skunkware. To use FSU Pthreads the application should include <pthread.h> and link with the "-lgthreads" linker option.

After running the `configure` script it is a good idea to examine the output which is stored in the file config.log. You may also need to edit the generated `config.h`, if any. Running `configure` with a tee to a log file

    ./configure | tee mout-config

will allow you to examine both the config.log and mout-config for possible errors in the configuration.

Many programs use the Imake system to configure the source for a particular platform. These will have an `Imakefile` which is used to generate the Makefile. To configure such a source hierarchy for compilation on your system simply run:

    xmkmf -a

Compiling Open Source Software

Once configured properly most Open Source software can be compiled by simply running "make". I use GNU make almost exclusively due to its widespread use in the Open Source community and the incompatibilities with System V make and different feature set. To be sure you are using GNU make rather than System V make, place `/usr/local/bin` prior to `/bin` or `/usr/bin` in your execution path. For example:

```
PATH=/usr/local/bin:$PATH
export PATH
```

To generate a logfile of the output from your compile, make can be piped to a tee as follows:

```
make 2>&1 | tee mout
```

After successfully compiling the source files and reviewing the logfiles many Makefiles include a "test" or "check" rule which runs any included automated tests. To run these, issue the command:

```
make test 2>&1 | tee mout-test
```

or

```
make check 2>&1 | tee mout-check
```

Finally, after compiling and testing the build, install the software with:

```
make install 2>&1 | tee mout-install
```

Forum presentation(s)

Modifying Open Source Software

When modifying Open Source software it is important to keep cross-platform portability in mind. The source you modify will in all likelihood be built on a wide variety of platforms. Your modifications cannot break either the build or functionality of the software on any other platform. It's also quite possible that your modifications could be useful to another platform. Thus, when possible, avoid platform-specific modifications like:

```
#if defined(__USLC__) /* UnixWare */
#define FOOBAR 1
#elif defined(_SCO_DS) /* OpenServer */
#define FOOBAR 0
#endif /* __USLC__ */
```

Whenever possible make such a modification based upon defines generated by the `configure` script. For example, `configure` might write out in the `config.h` file a define of something like HAVE_FOOBAR or NEED_FOOBAR. If so, a more portable construct for the same modification might look like:

```
#if defined(NEED_FOOBAR)
#define FOOBAR 1
#endif /* NEED_FOOBAR */
```

When making platform-specific modifications, the __USLC__ manifest define can be used for UnixWare and the _SCO_DS for OpenServer. If the application has been ported to Sun Solaris then it is often possible to simply change:

```
#if defined(SOLARIS)
```

to

```
#if defined(SOLARIS) || defined(__USLC__) || defined(_SCO_DS)
```

SCO OpenServer and UnixWare 7 are POSIX, XPG4, UNIX 95 and mostly UNIX 98 compliant. Often there are defines for these either in config.h or another header file. Most applications which comply with one or more of these standards will simply compile and link. Occasionally trivial modifications are necessary. For example, the location of the declaration of a function might reside in different header files and libraries on different systems. Usually the `man` command will indicate the syntax for inclusion of the declaration and any additional libraries to link against.

An excellent document on porting to UnixWare 7 is available at http://www.sco.com/forum1999/conference/develop/d16/. This document goes into these issues in much greater detail than is possible here.

Contributing to Open Source Projects

The Open Source development model relies upon contributions from vendors, developers, users and the primary development group. If you have successfully ported an Open Source project to a new platform or you have made source level modifications to enhance the software or fixed a bug, you can and should return your modifications to the project maintainers. In the case of ports and fixes for SCO platforms you may wish to first submit your modifications to the Skunkware development team for review and integration with other modifications to that package. To do so, e-mail your modifications to skunkware@sco.com.

Generally, modifications to an Open Source project are returned by e-mailing a context diff of the relevant changes. That is, if you modified source files `foo.c` and `bar.h` of project fubar release 1.0, you would create a context diff of these files:

```
diff -c foo.c.00 foo.c > mydiff-fubar-1.0
diff -c bar.h.00 bar.h » mydiff-fubar-1.0
```

This type of file difference is suitable for input to the patch utility. After creating the patch, test it on a fresh fubar 1.0 source tree. To return the modification, simply e-mail either the author of the program or, more commonly, the developer mailing list for the project. For instance:

```
mail fubar-dev@fubar.org
```

These mailing lists are an excellent means of staying abreast of the latest patches, releases, developer discussions, etc. Usually subscription is open or the list is archived for perusal by the public. Note that the above example is illustrative only. To determine the available mailing lists for a project, visit the project web site.

Contributing to SCO Skunkware Open Souce Software

If you would like to contribute to the ongoing effort to provide quality Open Source products to SCO customers:

- Read the Skunkware FAQ at http://www.sco.com/skunkware/faq.html
- Read the Skunkware submission guidelines at http://www.sco.com/skunkware/submission.html
- Join the polecats mailing list by sending an e-mail message to polecats-request@scofolks.ocston.org with any subject line and a single line in the body of the message:

    subscribe

The software on the Skunkware CD-ROM is licensed under a variety of terms. Much of it is licensed under the terms of the GNU General Public License. Some is licensed under the GNU Library General Public License. Other components are licensed under the Artistic License. Many of the components are "Freeware" with no restrictions on their redistribution while a few components are "Shareware" meaning the author would like you to try the software and, if you wish to use it, send her some money. A few components are commercial products which can be used freely for non-commercial purposes. Some components simply restrict their use to non-commercial purposes.

The Santa Cruz Operation, Inc. and SCO Skunkware are not related to, affiliated with or licensed by the famous Lockheed Martin Skunk Works (R), the creator of the F-117 Stealth Fighter, SR-71, U-2, Venturestar(tm), Darkstar(tm), and other pioneering air and spacecraft.

Author and Contributors

Ronald Joe Record has worked for The Santa Cruz Operation for over 16 years.

Record holds a Ph.D. in Mathematics from the University of California.

David Eyes (davidey@sco.com) contributed to this document in design, review and editorial matters.

About This Document

This document was created using SGML-Tools 2.0.2, Jade 1.2.1 and teTeX 0.9 running on UnixWare 7.

The source to this document is maintained at http://www.sco.com/skunkware/sgmldocs/scoworld-two.sgml. A Makefile and formatted varieties of this document are also available at http://www.sco.com/skunkware/sgmldocs/. For instance, you will find a postscript version at http://www.sco.com/skunkware/sgmldocs/ps/scoworld-two.ps. Finally, an HTML version of this and other porting documents will be available at http://www.sco.com/skunkware/porting/.