NAME

How to build SRecord

SPACE REQUIREMENTS

You will need about 3MB to unpack and build the *SRecord* package. Your milage may vary.

BEFORE YOU START

There are a few pieces of software you may want to fetch and install before you proceed with your installation of SRecord.

Boost Library

> You will need the C++ Boost Library. If you are using a package based system, you will need the libboost-devel package, or one named something very similar.
> http://boost.org/

Libgcrypt Library

> You will need the GNU Crypt library. If you are using a package based system, you will need the libgcrypt-devel package, or one named something very similar.
> http://directory.fsf.org/project/libgcrypt/

GNU Libtool

> You will need the GNU Libtool software, used to build shared libraries on a variety of systems.
> http://www.gnu.org/software/libtool/

GNU Groff

> The documentation for the *SRecord* package was prepared using the GNU Groff package (version 1.14 or later). This distribution includes full documentation, which may be processed into PostScript or DVI files at install time − if GNU Groff has been installed.

GCC You may also want to consider fetching and installing the GNU C Compiler if you have not done so already. This is not essential. SRecord was developed using the GNU C++ compiler, and the GNU C++ libraries.

The GNU FTP archives may be found at `ftp.gnu.org`, and are mirrored around the world.

SITE CONFIGURATION

The **SRecord** package is configured using the *configure* program included in this distribution.

The *configure* shell script attempts to guess correct values for various system-dependent variables used during compilation, and creates the *Makefile* and *lib/config.h* files. It also creates a shell script *config.status* that you can run in the future to recreate the current configuration.

Normally, you just *cd* to the directory containing *SRecord*'s source code and then type

```
% ./configure
```

...lots of output...

%

If you're using *csh* on an old version of System V, you might need to type

```
% sh configure
```

...lots of output...

%

instead to prevent *csh* from trying to execute *configure* itself.

Running *configure* takes a minute or two. While it is running, it prints some messages that tell what it is doing. If you don't want to see the messages, run *configure* using the quiet option; for example,

% ./configure −−quiet
%

To compile the **SRecord** package in a different directory from the one containing the source code, you must use a version of *make* that supports the *VPATH variable*, such as *GNU make*. *cd* to the directory where you want the object files and executables to go and run the *configure* script. *configure* automatically checks for the source code in the directory that *configure* is in and in .. (the parent directory). If for some reason *configure* is not in the source code directory that you are configuring, then it will report that it can't find the

source code. In that case, run *configure* with the option `--srcdir=`*DIR*, where *DIR* is the directory that contains the source code.

By default, *configure* will arrange for the *make install* command to install the **SRecord** package's files in */usr/local/bin*, and */usr/local/man*. There are options which allow you to control the placement of these files.

`--prefix=`*PATH*
: This specifies the path prefix to be used in the installation. Defaults to */usr/local* unless otherwise specified.

`--exec-prefix=`*PATH*
: You can specify separate installation prefixes for architecture-specific files files. Defaults to *${prefix}* unless otherwise specified.

`--bindir=`*PATH*
: This directory contains executable programs. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/bin* unless otherwise specified.

`--mandir=`*PATH*
: This directory contains the on-line manual entries. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/man* unless otherwise specified.

configure ignores most other arguments that you give it; use the `--help` option for a complete list.

On systems that require unusual options for compilation or linking that the *SRecord* package's *configure* script does not know about, you can give *configure* initial values for variables by setting them in the environment. In Bourne-compatible shells, you can do that on the command line like this:

```
$ CXX='g++ -traditional' LIBS=-lposix ./configure
...lots of output...
$
```

Here are the *make* variables that you might want to override with environment variables when running *configure*.

Variable: CXX
: C++ compiler program. The default is *c++*.

Variable: CPPFLAGS
: Preprocessor flags, commonly defines and include search paths. Defaults to empty. It is common to use `CPPFLAGS=-I/usr/local/include` to access other installed packages.

Variable: INSTALL
: Program to use to install files. The default is *install* if you have it, *cp* otherwise.

Variable: LIBS
: Libraries to link with, in the form `-l`*foo* `-l`*bar*. The *configure* script will append to this, rather than replace it. It is common to use `LIBS=-L/usr/local/lib` to access other installed packages.

If you need to do unusual things to compile the package, the author encourages you to figure out how *configure* could check whether to do them, and mail diffs or instructions to the author so that they can be included in the next release.

BUILDING SRECORD

All you should need to do is use the

```
% make
...lots of output...
%
```

command and wait. When this finishes you should see a directory called *bin* containing three files: *srec_cat*, *srec_cmp* and *srec_info*.

srec_cat *srec_cat* program is used to manipulate and convert EPROM load files. For more information, see *srec_cat*(1).

srec_cmp
The *srec_cmp* program is used to compare EPROM load files. For more information, see *srec_cmp*(1).

srec_info
The *srec_info* program is used to print information about EPROM load files. For more information, see *srec_info*(1).

If you have GNU Groff installed, the build will also create a *etc/reference.ps* file. This contains the README file, this BUILDING file, and all of the man pages.

You can remove the program binaries and object files from the source directory by using the

```
% make clean
...lots of output...
%
```

command. To remove all of the above files, and also remove the *Makefile* and *lib/config.h* and *config.status* files, use the

```
% make distclean
...lots of output...
%
```

command.

The file *etc/configure.in* is used to create *configure* by a GNU program called *autoconf*. You only need to know this if you want to regenerate *configure* using a newer version of *autoconf*.

Windows NT
It is possible to build SRecord on MS Windows platforms, using the Cygwin (see `www.cygwin.com`) or DJGPP (see `www.delorie.com/djgpp`) environments. This provides the "porting layer" necessary to run Unix programs on Windows. The build process is exactly as described above.

You may need to pass in the include path to the Boost library. This is most simply done as

```
CC='gcc -no-cygwin' \
CXX='g++ -mno-cygwin -I/usr/include/boost-1_33_1' \
```

DJGPP always produces native binaries, however if you want to make native binaries with Cygwin (*i.e.* ones which work outside Cygwin) there is one extra step you need after running `./configure` and before you run `make`. You need to edit the *Makefile* file, and add `-mno-cygwin` to the end of the `CXX=g++` line.

Once built (using either tool set) Windows binaries should be testable in the same way as described in the next section. However, there may be some CRLF issues in the text file comparisons which give false negatives, depending on the CRLF setting of your Cygwin file system when you unpacked the tarball.

TESTING SRECORD

The *SRecord* package comes with a test suite. To run this test suite, use the command

```
% make sure
...lots of output...
Passed All Tests
%
```

The tests take a few seconds each, with a few very fast, and a couple very slow, but it varies greatly depending on your CPU.

If all went well, the message

```
Passed All Tests
```

should appear at the end of the make.

INSTALLING SRECORD

As explained in the *SITE CONFIGURATION* section, above, the *SRecord* package is installed under the */usr/local* tree by default. Use the `--prefix=`*PATH* option to *configure* if you want some other path. More specific installation locations are assignable, use the `--help` option to *configure* for details.

All that is required to install the *SRecord* package is to use the

```
% make install
...lots of output...
%
```

command. Control of the directories used may be found in the first few lines of the *Makefile* file and the other files written by the *configure* script; it is best to reconfigure using the *configure* script, rather than attempting to do this by hand.

GETTING HELP

If you need assistance with the *SRecord* package, please post to the srecord-users mailing list

```
srecord-users@lists.sourceforge.net
```

For information obout the srecord-users mailing list. http://srecord.sourceforge.net/mailing-list.html

When reporting problems, please include the version number given by the

```
% srec_cat -version
srecord version 1.64.D001
...warranty disclaimer...
%
```

command. Please do not send this example; run the program for the exact version number.

COPYRIGHT

srecord version 1.64

Copyright © 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014 Peter Miller
Copyright © 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014 SCott Finneran

The *SRecord* package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

It should be in the *LICENSE* file included with this distribution.

AUTHOR

Scott Finneran E-Mail: scottfinneran@yahoo.com.au
Peter Miller E-Mail: pmiller@opensource.org.au