NAME

How to build libexplain

SPACE REQUIREMENTS

You will need about 6MB to unpack and build the *libexplain* 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 libexplain

libcap Linux needs libcap, for access to capabilities.
ftp://ftp.kernel.org/pub/linux/libs/security/linux−privs/kernel−2.2/

lsof

For systems with inadequate or non-existent /proc facilities, and that includes *BSD and MacOS X, the *lsof*(1) program is needed to obtain supplementary information about open file descriptors. However, if *lsof*(1) is not supported on your operating system, libexplain will still work, but some useful information (such as translating file descriptors into the name of the open file) will be absent from error explanations.

ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/
http://people.freebsd.org/~abe/

You **must** have *lsof*(1) installed on *BSD and Solaris, otherwise the test suite will generate staggering numbers of false negatives. It will produce less informative error messages, too.

Supported systems include: Free BSD, HP/UX, Linux, Mac OS X, NetBSD, Open BSD, Solaris, and several others.

GNU libtool

The libtool program is used to build shared libraries. It understands the neccesary, weird and wonderful compiler and linker tricks on many weird and wonderful systems.
http://www.gnu.org/software/libtool/

bison The bison program is a general-purpose parser generator that converts a grammar description for an LALR(1) context-free grammar into a C program to parse that grammar.
http://www.gnu.org/software/bison/

GNU Groff

The documentation for the *libexplain* 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. libexplain 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 **libexplain** 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 *libexplain/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 *libexplain*'s source code and then type

```
$ ./configure --prefix=/usr
...lots of output...
$
```

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

```
% sh configure --prefix=/usr
...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 −−prefix=/usr −−quiet
$

To compile the **libexplain** 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. The *configure* script automatically checks for the source code in the directory that *configure* is in and in .IR .. (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 **libexplain** package's files in */usr/local/bin*, */usr/local/lib*, */usr/local/include*, 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 *libexplain* 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:

$ `CC='gcc −ansi' LIBS=−lposix ./configure`
...lots of output...
$

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

Variable: CC
: C compiler program. The default is *gcc*.

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*(1) if you have it, *cp*(1) 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 LIBEXPLAIN

All you should need to do is use the

```
$ make
...lots of output...
$
```

command and wait. This can take a long time, as there are a few thousand files to be compiled.

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 *libexplain/config.h* and *config.status* files, use the

```
$ make distclean
...lots of output...
$
```

command.

The file *etc/configure.ac* 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*.

TESTING LIBEXPLAIN

The *libexplain* 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 fraction of a second each, with most 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.

Sources of False Negatives

There are a number of factors that can cause tests to fail unnecessarily.

Root
: You will get false negatives if you run the tests as root.

Architecture
: Some errors move around depending on architecture (sparc *vs* x86 *vs* s390, *etc*). Some even move around due to different memory layout for 32-bit *vs* 64-bit, for the same processor family. For example, when testing EFAULT explanations.

strerror
: Different systems have different *strerror*(3) implementations (the numbers vary, the texts vary, the existence varies, *etc*). This can even be incompatible across Linux architectures when ABI compatibility was the goal, *e.g.* sparc *vs* i386.

ioctl
: There are (at least) three inconsistent implementations of ioctl request macros, all incompatible, depending on Unix vendor. They also vary on Linux, depending on architecture, for ABI compatibility reasons.

Environment
: Some tests are difficult because the build-and-test environment can vary widely. Sometimes it's a chroot, sometimes it's a VM, sometimes it's fakeroot, sometimes it really is running as root. All

these affect the ability of the library to probe the system looking for the proximal cause of the error, *e.g.* ENOSPC or EROFS. This often results in 2 or 4 or 8 explanations of an error, depending on what the library finds, *e.g.* existence of useful information in the mount table, or not.

Mount Table
: If you run the tests in a chroot jail build environment, maybe with bind mounts for the file systems, it is necessary to make sure */etc/mtab* (or equivalent) has sensable contents, otherwise some of the path resolution tests will return false negatives.

/proc
: If your system has a completely inadequate `/proc` implementation (including, but not limited to: *BSD, Mac OS X, and Solaris) or no `/proc` at all, **and** you have not installed the *lsof*(1) tool, then large numbers of tests will return false negatives.

As these problem have occured, many of the tests have been enhanced to cope, but not all false negative situations have yet been discovered.

INSTALLING LIBEXPLAIN

As explained in the *SITE CONFIGURATION* section, above, the *libexplain* 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 *libexplain* 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.

Note: if you are doing a manual install (as opposed to a package build) you will also need to run the

```
# ldconfig
#
```

command. This updates where the system thinks all the shared libraries are. And since we just installed one, this is a good idea.

GETTING HELP

If you need assistance with the *libexplain* package, please do not hesitate to contact the author at

```
Peter Miller <pmiller@opensource.org.au>
```

Any and all feedback is welcome.

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

```
$ explain -version
explain version 1.4.D001
...warranty disclaimer...
$
```

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

COPYRIGHT

libexplain version 1.4

Copyright © 2008, 2009, 2010, 2011, 2012, 2013, 2014 Peter Miller

The *libexplain* 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 Lesser General Public License for more details.

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

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/