FHist

Reference Manual

Peter Miller
pmiller@opensource.org.au

.

This document describes FHist version 1.18
and was prepared 9 October 2019.

This document describing the FHist package, and the FHist package itself, are
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

NAME

fhist – file history and comparison tools

Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

Portions of this program are
Copyright © 1990 David I. Bell.

The *fhist* package is distributed under the terms of the GNU General Public License, see the *LICENSE* section, below, for more information.

DESCRIPTION

The FHist package contains 3 utilities, a file history tool "*fhist*", a file comparison tool "*fcomp*", and a file merging tool "*fmerge*". All three are bundled together, because they all use the same minimal-difference algorithm.

fhist
: Keeps track of versions of a file. It works correctly when given binary files as input. See *fhist*(1) for more information.

fcomp
: Compares two versions of a file, usually line-for-line textual comparison. It is capable of comparing two binary files byte-for-byte. See *fcomp*(1) for more information.

fmerge
: Merges together edits from two descendants of a file. See *fmerge*(1) for more information.

The history tool presented here, fhist, is a *minimal* history tool. It provides no locking or branching. This can be useful in contexts where the configuration management or change control be being provided by some other tool.

REFERENCES

This program is based on the algorithm in

An O(ND) Difference Algorithm and Its Variations, Eugene W. Myers, TR 85-6, 10-April-1985, Department of Computer Science, The University of Arizona, Tuscon, Arizona 85721.

See also:

A File Comparison Program, Webb Miller and Eugene W. Myers, Software Practice and Experience, Volume 15, No. 11, November 1985.

BUILDING

For complete instructions for host to build these programs, see the *BUILDING* file included in this distribution.

ARCHIVE SITE

The latest version of *fhist* is available on the Web from:

```
URL:    http://fhist.sourceforge.net/
File:   index.html          # The FHist page.
File:   fhist-1.18.README   # Description, from the tar file
File:   fhist-1.18.lsm      # Description, in LSM format
File:   fhist-1.18.spec     # RedHat package spec
File:   fhist-1.18.tar.Z    # The complete source.
```

FHist is also carried by sunsite.unc.edu in its Linux archives. You will be able to find FHist on any of its mirrors.

```
URL:    ftp://sunsite.unc.edu/pub/Linux/devel/vc/
File:   fhist-1.18.README   # Description, from the tar file
File:   fhist-1.18.lsm      # Description, in LSM format
File:   fhist-1.18.spec     # RedHat package spec
File:   fhist-1.18.tar.Z    # The complete source.
```

This site is extensively mirrored around the world, so look for a copy near you (you will get much better response).

COPYRIGHT

fhist version 1.18.D001
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is derived from a work
Copyright © 1990 David I. Bell.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

AUTHORS

Peter Miller	Web:	http://miller.emu.id.au/pmiller/
/\/*	E-Mail:	pmiller@opensource.org.au
David I. Bell	Web:	http://www.canb.auug.org.au/~dbell
	E-Mail:	dbell@canb.auug.org.au

RELEASE NOTES

For excruciating detail, and also acknowledgements of those who generously sent me feedback, please see the *etc/CHANGES.1.18* file included in this distribution.

A number of features and bug fixes have been added to *fhist* with this release. A few of them are detailed here:

Version 1.18 (16-Oct-2009)

• The .XX macro usage has been made conditional in the man pages, to silence Debian lintian warnings.

• The use of naked "-" have been replaced with "\-" for option introducers, "\[hy]" for hyphens, and "\[mi]" for minus signs, to silence Debian lintian warnings.

• The author's email address has been updated.

Version 1.17 (17-Jun-2008)

• The licenseis now GPLv3.

• The *fhist*(1) command now accepts a remark string on the command line.

Version 1.16 (20-Dec-2005)

• There is a new *fmerge −ignore-identical-conflicts* option which may be used to suppress logical conflicts in which the same thing is done by both variants. This is often a better match for users' expectations for merging source code.

Version 1.15 (8-Nov-2005)

• There is a new *fcomp −no-binary* option, which may be used to prevent the comparison of binary files, instead it treats them both as empty.

• A small build problem on MacOS X has been fixed.

Version 1.14 (8-Jun-2004)

• The ./configure script now understands the --with-nlsdir option, so that you can place the `.mo` files.

• The *fhist*(1) program is now able to cope with numeric modules names.

• The occasional false negative from test 26 has been fixed. It was failing for some users because of message translation (internationalization) issues.

Version 1.13 (13-Mar-2003)

• A bug has been fixed in some of the tests. They were susceptible to false negatives if the display width changed.

• All references to the cuserid function have been replaced. It isn't sufficiently portable to be used in real programs.

Version 1.12 (28-Nov-2002)

• Some build problems have been fixed.

Version 1.11 (26-Nov-2002)

• Some build problems, relating to modern ANSI C compilers choking on K&R function definitions with variable arguments, have been fixed.

• Two bugs relating to the handling of binary files by *fhist*(1) have been fixed.

• A bug which left garbage files behind when a create failed has been fixed.

Version 1.10 (9-Jul-2002)

• Interrupt handling has been improved.

• There is a new *fhist -No_Keywords* option, used to completely disable keyword substitution.

• Several build problems have been fixed.

Version 1.9 (23-Oct-2001)
No public release.

Version 1.8 (16-Oct-2001)
• There is a new **−BINary** option for the *fcomp*(1) program, which compares binary files a byte at time, printing the results in hexadecimal.

• The *fcomp*(1) program now silently copes with CRLF line terminations.

Version 1.7 (11-Apr-2000)
• The *fhist*(1) command now has a **−binary** option, which may be used to store the history of binary files.

• The *fhist*(1) command has a new **−make-path** option, which requests that the history directory be created if necessary.

• A bug in *fhist*(1) wich caused a SEGFAULT when you used the **−t** option (extract to terminal) has been fixed.

Version 1.6 (25-Oct-1999)
• An RPM spec file has been added to the distribution.

• The code is now more robust about what various UNIX systems return from pathconf().

• A bug with the "*fcomp -blank*" option has been fixed.

Version 1.5 (1-Jun-1999)
• Binary files are now detected on input, and the utilities file gracefully with a warning or error message, as appropriate.

• Some buffer over-run bugs have been fixed.

• Several improvements have been made to the portability.

Version 1.4 (16-Sep-1998)
• The install and build procedures have been made more robust, and they take note of more of the information provided by GNU Autoconf.

• The error messages have been internationalized, so it is now possible to obtain error messages in your native language. (If you would like to contribute with error message translations, please contact the author.)

• An LSM description has been added, along with a HTML page to present it all nicely at the archive site.

• A RedHat Package Manager spec file has been added, so that a RedHat package can be created. The spec file is in the standard distribution.

Version 1.3 (29-Mar-1998)
This version was not distributed at all.

Version 1.2

This version was not distributed very widely.

• The non-standard isblank function is no longer used, it cause too many portability problems.

• The use of pathconf is not more robust for more operating systems.

Version 1.1

• The *fhist* package now uses a shell script called *configure* to configure itself. This script is generated using the *GNU Autoconf* utility. This should make *fhist* significantly easier to configure, and significantly more portable.

• A bug has been fixed in the conflict reporting of the *fmerge* program. It now correctly opens the conflicts file.

• The *fhist* program now uses *pathconf*(2) to determine file name length limits.

NAME

fhist – file history and comparison tools

Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller

Portions of this program are
Copyright © 1990 David I. Bell.

The *fhist* package is distrributed under the terms of the GNU General Public License, see the *LICENSE* section, below, for more information.

SPACE REQUIREMENTS

You will need about 600K to unpack and build the *fhist* package. (This is the worst case seen so far, most systems have binaries about 60% as big as this, 400K is more typical.) Your mileage 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 cook.

GNU Gettext
: The *fhist* package has been internationalized. It can now print error messages in any of the supported languages. In order to do this, the GNU Gettext package must be installed *before* you run the configure script as detailed in the next section. This is because the configure script looks for it. On systems which use the GNU C library, version 2.0 or later, there is no need to explictly do this as GNU Gettext is included. Remember to use the GNU Gettext configure *--with-gnu-gettext* option if your system has native gettext tools.

GNU Groff
: The documentation for the *fhist* package was prepared using the GNU Groff package. This distribution includes full documentation, which may be processed into PostScript or DVI files at install time – if GNU Groff has been installed.

Bison
: If your operating system does not have a native *yacc*(1) you will need to fetch and install GNU Bison in order to build the *fhist* package.

An ANSI C compiler.
: You must have an ANSI C compiler to compile this program. You may also want to consider fetching and installing the GNU C Compiler if you have not done so already.

The GNU FTP archives may be found at `prep.ai.mit.edu`, and are mirrored around the world.

SITE CONFIGURATION

The **fhist** package is configured using the *configure* shell script 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 *common/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 *fhist*'s source code and 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* with its standard output redirected to */dev/null*; for example,

```
% ./configure > /dev/null
```

```
%
```

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

`--prefix=`*PATH*
: You can specify an installation prefix other than */usr/local* by giving *configure* the option `--prefix=`*PATH*.

`--exec-prefix=`*PATH*
: You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you give *configure* the option `--exec-prefix=`*PATH* the **fhist** package will use *PATH* as the prefix for installing programs and libraries. Data files and documentation will still use the regular prefix. Normally, all files are installed using the same prefix.

`--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.

`--datadir=`*PATH*
: This directory contains installed data, such as the documentation, reports and shell scripts distributed with Aegis. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/share/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--libdir=`*PATH*
: This directory contains installed data, such as the error message catalogues. 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}/lib/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--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.

`--with-nlsdir=`*PATH*
: This directory contains machine specific message catalogues. Defaults to `--libdir` if not set.

configure ignores any other arguments that you give it.

On systems that require unusual options for compilation or linking that the *fhist* 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 -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: CC
: C compiler program. The default is *cc*.

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.

Variable: NLSDIR
This works the same way as the `--with-nlsdir` option.

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 FHIST

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 four files: *fcomp*, *fhist*, *fmerge* and *txt2c*.

fcomp The *fcomp* program is user to compare two text files.

fhist The *fhist* program is used to maintain and edit history of a text file.

fmerge The *fmerge* program is used to merge together edits from two descendants of a file.

txt2c The *txt2c* program is a utility used to build the *fhist* package; it is not intended for general use and should not be installed.

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 *common/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*.

TESTING FHIST

The *fhist* package is accompanied by a test suite. To run this test suite, use the following command:

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

This is successful if the last line of the test output reads "Passed All Tests".

Please let the author know if any of the tests fail, and why if you can work that out.

INSTALLING FHIST

As explained in the *SITE CONFIGURATION* section, above, the *fhist* package is installed under the */usr/local* tree by default. Use the `--prefix=`*PATH* option to *configure* if you want some other path.

All that is required to install the *fhist* 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 if you want to bypass the *configure* script.

The above procedure assumes that the *soelim*(1) command is somewhere in the command search *PATH*. The *soelim*(1) command is available as part of the *GNU Roff* package, mentioned previously in the *PRINTED MANUALS* section. If you don't have it, but you do have the *cook* package, then a link from *roffpp* to *soelim* will also work.

The above procedure also assumes that the *$(prefix)/man/man1* and *$(prefix)/man/man5* directories already exist. If they do not, you will need to *mkdir* them manually.

PRINTED MANUALS

The easiest way to get copies of the manuals is to get the *fhist.1.18.pdf* file from the archive site. This is an Adobe AcroRead file containing the Reference Manual, which contains the README file, the BUILDING file and internationalization notes, as well as all of the manual pages for all of the commands.

This distribution contains the sources to all of the documentation for *fhist*. The author used the GNU groff package and a postscript printer to prepare the documentation. If you do not have this software, you will need to substitute commands appropriate to your site.

If you have the GNU Groff package installed *before* you run the *configure* script, the *Makefile* will contain instructions for constructing the documentation. If you already used the *make* command, above, this has already been done. The following command

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

can be used to do this explicitly, if you managed to get to this point without doing it. Please note that there may be some warnings from groff, particularly about missing fonts, particularly for the `.txt` files; this is normal.

Once the documents have been formatted, you only need to print them. The following command

% `lpr lib/en/reference.ps`
%

will print the English PostScript version of the Reference Manual. Watch the *make* output to see what other versions are available.

COPYRIGHT

fhist version 1.18.D001
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is derived from a work
Copyright © 1990 David I. Bell.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

AUTHORS

Peter Miller	Web:	http://miller.emu.id.au/pmiller/
/\/*	E-Mail:	pmiller@opensource.org.au
David I. Bell	Web:	http://www.canb.auug.org.au/~dbell
	E-Mail:	dbell@canb.auug.org.au

NAME

Internationalization

DESCRIPTION

The FHist package has gone international; it can now speak many languages. This is accomplished by using the GNU Gettext library and utilities. In order to do this, is is necessary to install GNU Gettext prior to configuring, making and installing the FHist package, as described in the *BUILDING* file.

Internationalization

This is the process of identifying all of the error messages in the source code, and providing error message catalogues in a variety of languages. The error message identification was performed by the FHist package's author, and the various GNU translation teams provided the translations. Users of the FHist package do not need to do anything to internationalize it, this has already been done.

Localization

The programs in the FHist package are "localizable" when properly installed; the programs they contain can be made to speak your own native language.

By default, the FHist package will be installed to allow translation of messages. It will automatically detect whether the system provides a usable 'gettext' function.

INSTRUCTIONS FOR USERS

As a user, if your language has been installed for this package, you only have to set the 'LANG' environment variable to the appropriate ISO 639 two-letter code prior to using the programs in the package. For example, let's suppose that you speak German. At the shell prompt, merely execute

```
setenv LANG de
```

(in 'csh'), or

```
LANG=de; export LANG
```

(in 'sh'). This can be done from your *.cshrc* or *.profile* file, setting this automatically each time you login.

An operating system might already offer message localization for many of its programs, while other programs have been installed locally with the full capabilities of GNU Gettext. Using the GNU Gettext extended syntax for the 'LANG' environment variable may break the localization of already available through the operating system. In this case, users should set both the 'LANGUAGE' and 'LANG' environment variables, as programs using GNU Gettext give preference to the 'LANGUAGE' environment variable.

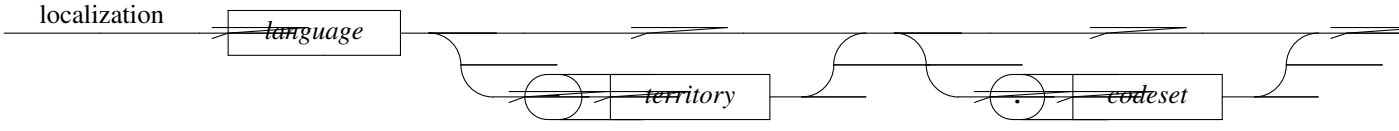
For example, some Swedish users would rather read translations in German when Swedish is not available. This is easily accomplished by setting 'LANGUAGE' to 'sv:de' while leaving 'LANG' set to 'sv'.

DIRECTORY STRUCTURE

All files which may require translation are located below the *lib* directory of the source distribution. It is organized as one directory below *lib* for each localization. Localizations include all documentation as well as the error messages.

Localization Directory Names

Each localization is contained in a sub-directory of the *lib* directory, with a unique name. Each localization sub-directory has a name of the form:



language is one of the 2-letter names from the ISO 639 standard. See *http://www.ics.uci.edu/pub/ietf/-http/related/iso639.txt* for a list.

territory is one of the 2-letter country codes from the ISO 3166 standard. See *ftp://rs.internic.net/-netinfo/iso3166-countrycodes* for a list.

codeset is one of the codeset names defined in RFC 1345. This simplifies the task of moving localizations between charsets, because GNU Recode understands them. See *http://info.internet.isi.edu/1s/in-notes/rfc/files/rfc1345.txt* for a list.

Here are some examples of localization names:

Name	Description
en.ascii	English, ASCII encoding
en_us.ascii	English with US spelling
de.latin1	German, Latin-1 encoding

Localization Directory Contents

Each localization sub-directory in turn contains sub-directories. These are:

Directory	Contents
LC_MESSAGES	The error message (PO) files
building	The BUILDING file
man1	The section 1 manual entries
readme	The README file
building	The BUILDING file
reference	The Reference Manual

The structure is identical below each of the localization directories. The sub-directories of all localizations will have the same names.

INSTRUCTIONS FOR TRANSLATORS

When translating the error messages, all of the substitutions described in *cook_sub*(5) are also available. Substitution variable names and function names may be abbreviated, in the same way that command line options are abbreviated, but abbreviation should probably be avoided. Substitution names will *never* be internationalized, otherwise the substitutions will stop working, Catch-22.

While FHist was written by an English speaker, the English localization is necessary, to translate the "terse programmer" style error messages into something more user friendly.

Messages which include command line options need to leave the command line options untranslated, because they are not yet internationalized, though they may be one day.

Each LC_MESSAGES directory within each localization contains a number of PO files. There is one for each program in the FHist package, plus one called `common.po` containing messages common to all of them. The MO file for each program is generated by naming the program specific PO file and also the common PO file.

NAME

fcomp – file compare

SYNOPSIS

fcomp [*option...*] *filename1 filename2*

fcomp –Help

fcomp –VERSion

DESCRIPTION

The *fcomp* program is used to compare text files, similar to the *diff*(1) program. Its advantage is that it always produces minimal differences, and so will never mis-sync when comparing files. Its disadvantage is that it runs slower due to the extra work required to produce optimal differences. However, for files differing by less than a few thousand lines, its performance is adequate. The algorithms used by this utility are also used by the *fhist*(1) program in order to produce the edit history.

To compare file *old* to file *new*, the command:

fcomp old new

would be used. This gives the differences involved in converting **from** file *old* **to** file *new*. This is analogous to the use of the *cp*(1) command. Either the *old* or *new* file may be a directory, in which case the comparison is done to the file in the directory with the same name as the other file. An error is given if *old* and *new* are both directories.

OPTIONS

The following options are understood:

–BINary

This option may be used to compare binary files on a byte-for-byte basis. (Each byte is treated as a “line” by the algorithm.) Byte values are displayed in hexadecimal, as are the addresses. Note: this is different behaviour to the *fhist*(1) option of the same name.

–No_BINary

This option may be used to avoid comparing binary files. A warnign will be prointed on the standard error, but the program will report success without printing andy other output.

–Blank

Ignore blank lines in the input files.

–Context *number*

This specifies the number of lines of "context" which is displayed. This shows the specified number of lines before and after the actual lines being changed. This is useful to locate and identify the line which is actually being changed, when there are many identical copies of the line in the file.

–Edit

Output an edit script which is machine readable.

–Failures *number*

This stops the comparison if the number of changes exceeds the specified number. Each change is a delete or insert of a single line. This is useful when you are not interested in the results when the files are totally different. Another use is a quick check to see if two files are identical, by using a value of zero.

–Help

Give some help on how to use the *fcomp* program.

–Join *number*

This merges together lines which have changed, if they are separated by up to the specified number of unchanged lines. This makes a change look bigger, but reduces the "choppiness" of the output by showing fewer regions being changed. This is particularly effective to suppress

worthless matchings of single blank lines or comment beginning and ending lines. A useful value for this option is 3 or so.

−Matching
Output matching lines, rather then changed lines.

−Number
This outputs the line numbers at the left edge of the output. This isn't normally needed, since the line numbers are displayed in the comment line preceding the lines being displayed. Not outputting the line numbers prevents the terminal from needlessly scrolling for long lines.

−Output *filename*
Send the output to this file, rather than the standard output.

−Quiet
Output only a quick summary of changes needed.

−Spaces
This option ignores differences in the number of spaces in the two lines. That is, two or more adjacent spaces are handled as a single space. Spaces at the beginning or end of a line are totally ignored.

−Upcase
Uppercase lines before comparing.

−VERSion
Show what version of *fcomp* is running.

−What
This outputs all of both files together, showing what happened to each line of the first file in order to change to the line in the second file. This output is in "change bar" format, where inserted lines begin with **|+**, deleted lines begin with **|−**, and unchanged lines begin with spaces. The presence of the vertical bar makes it easy to search for the changed lines.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "−help, "−HELP" and "−h" are all interpreted to mean the **−Help** option. The argument "−hlp" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line.

The GNU long option names are understood. Since all option names for *fcomp* are long, this means ignoring the extra leading '−'. The "**−***option***=***value*" convention is also understood.

FILE NAME EXPANSION

As a convenience, if a pathname begins with a period and a environment variable exists with that name, then the value of the environment variable will be used as the actual pathname. For example, if a environment variable of *.FOO* has the value *this.is.a.long.name*, then the command

fcomp −o .FOO

is actually equivilant to the command

fcomp −o this.is.a.long.name

If you want to prevent the expansion of a pathname which begins with a period, then you can use an alternate form for the pathname, as in:

fcomp −o ./.FOO

BINARY FILES

In general, fcomp can handle all text files you throw at it, even international text with unusual encodings. However, fcomp is *unable* to cope elegantly with files which contain the NUL character.

The *fcomp*(1) program simply prints a warning, and continues, you need to know that it converts NUL characters into an 0x80 value before performing the comparison.

The *fmerge*(1) program also converts the NUL character to an 0x80 value before merging, after a warning, and any output file will contain this value, rather than the original NUL character.

The *fhist*(1) program, however, generates a fatal error if any input file contains NUL characters. This is intended to protect your source files for unintentional corruption. Use **−BINary** for files which absolutely must contain NUL characters.

EXIT STATUS

The *fcomp* program will exit with a status of 1 on any error. The *fcomp* program will only exit with a status of 0 if there are no errors.

REFERENCES

This program is based on the algorithm in

An O(ND) Difference Algorithm and Its Variations, Eugene W. Myers, TR 85-6, 10-April-1985, Department of Computer Science, The University of Arizona, Tuscon, Arizona 85721.

See also:

A File Comparison Program, Webb Miller and Eugene W. Myers, Software Practice and Experience, Volume 15, No. 11, November 1985.

COPYRIGHT

fcomp version 1.18.D001
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is derived from a work
Copyright © 1990 David I. Bell.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

AUTHORS

Peter Miller Web: http://miller.emu.id.au/pmiller/
/\/* E-Mail: pmiller@opensource.org.au

David I. Bell Web: http://www.canb.auug.org.au/~dbell
E-Mail: dbell@canb.auug.org.au

NAME

fhist – file history

SYNOPSIS

fhist *filename... option...*

fhist –Help

fhist –VERSion

DESCRIPTION

The *fhist* program is used to keep track of the successive versions of a file. Using this program, you can remember all of your changes to a file, and get back any one of the old versions. The uses of this ability are:

1. You can make a series of tentative edits to the file, and if necessary back up to the last "good" edit.
2. You can delete old subroutines and code from your file which are obsolete, but still be able to get them back in the future in case a need for them arises.
3. You can compare two versions of the file to see how you fixed some old problem, so that you can check up on the correctness of the fix at a later date.
4. You get a record of your remarks for each version, so that you can quickly know what bugs were fixed, and what features were implemented.
5. The date the file was last edited can be automatically stored in the file.

The *fhist* program manipulates modules. A module is simply any text file that you are interested in keeping versions of. For example, a source file *doit.c* is a module, and so is a documentation file *howto.doc*. The module name includes the suffix of the file (as in the above examples). However, pathnames are not part of a module name, so that */usr/dbell/bar.c* cannot be a legal module name. A module name is limited to 12 characters since the *fhist* program needs two extra characters for its own purpose.

Keyword Substitution

It is possible to have information about the state of the file inserted into the file. See the **–Modify** and **–No-Keywords** options, below, for more infromation.

OPTIONS

The following options are understood:

–Path *pathname*

Modules are stored in a directory, called the module storage directory. The default directory is *FHIST*, and therefore is located relative to your current directory. This is convenient when you are in a directory containing many modules, and you want a local storage directory to contain just those modules. If you use the **–p** option, then you can locate the storage directory anywhere you choose. This is useful if you choose to have a common storage directory for all of your files, independent of where they actually are used.

The files inside of the storage directory should not be changed by you. Doing so will probably corrupt your edit history, causing errors when you extract old revisions. For your information, though, each module is stored as two files in the directory. The one with the *.s* suffix is a copy of the newest version of the module, with one extra line at the beginning. The one with the *.e* suffix is the edit history of the module, and contains the information needed to extract previous revisions of the module. Thus if the edit history is ever corrupted, you will at least have the most recent version of the module.

–MaKe_Path

This option may be used to request that the path directory be created automatically if it does not yet exist. This works for both the directory specified by the –Path option, and for the default.

Intermediate directories will also be created if necessary.

−BINary

This option may be used to specify that the file is binary, that it may contain NUL characters. It is essential that you have consistent presence or absence of the **−BINary** option for each file when combined with the **−CReate**, **−Update**, **−Conditional_Update** and **−Extract** options. Failure to do so will produce inconsistent results. Note: this is different behaviour to the *fcomp*(1) option of the same name. Note: the **−BINary** option does *not* imply the **−No-Keywords** option.

−CReate

To use the *fhist* program for the first time, you need to create your storage directory. Therefore, *cd* to the directory where you want it to be, which is probably the directory containing the modules you want to save the revisions of. Then create the directory *FHIST* (or some other name if you don't want to use the default name).

To start using a module under *fhist*, you must first use the **−CReate** option. This creates the initial edit for that module in the storage directory, with the contents of the specified module as the initial edit. Thus, if you have a source file *prog.c*, then the command:

fhist prog.c −create

creates the initial edit of the module. As part of this process, you are asked to provide remarks about the file. These remarks can be seen later using the **−List** option (described below). After the remarks have been typed, the contents of the file are then saved. You can then delete the file *prog.c* if desired, and *fhist* would be able to recreate it later. Or you can leave it there as the working copy of the module.

The **−CReate** option may be combined with the **−Update** or **−Conditional_Update** options to create the file if required.

−Update

To save another revision of the module, you use the **−Update** option. This updates the files in the storage directory to include the latest changes. Remarks are again asked for so that you can document why you made this edit. Thus, to continue the example, after editing *prog.c*, the command:

fhist prog.c −u

will save the changes as a new edit. This command compares the newest version of the module to the previous version, saves the differences in the *.e* file, and copies the new source to the *.s* file. At this point, you can once again delete the *prog.c* file if desired, and later get back either of the two versions of the program.

The *fhist* program handles quota or disk full problems during a create or update operation without damage occurring to the edit history files. If an edit cannot be completed because of such problems, the edit is backed out completely, and you will get an error message about the disk problem. There is no need for any error recovery in this case, other than retrying the update when more disk space is available. The *fhist* program also disables signals during the critical file operations, so you do not have to worry about damaging the edit history files because of attempts to quit out of the program.

The **−CReate** option may be combined with the **−Update** or **−Conditional_Update** options to create the file if required.

−Input *filename*

In either the **−CReate** or **−Update** options, the file containing the new version of the module defaults to the same name as the module. In the example, the module *prog.c* was created and updated from the data in the file *prog.c*. When you wish the data to come from some other file, you can use the **−Input** option, which specifies the input file to use for the data. For example, if you wanted to update *prog.c*, but from a filename called *newprog.c*, then the command:

fhist prog.c −u −i newprog.c

would save a new revision of module *prog.c*, but with the data that was in the file *newprog.c*. In this case, the file *prog.c* does not have to exist, and isn't referenced even if it did exist. Again,

once the update is complete, you could delete the *newprog.c* file if desired and then later you can retrieve its contents.

−Remarks

Remarks can be read from a file instead of from the terminal. The **−Remarks** option can be used to specify a file name containing the remarks. If there is no file name following the **−Remarks** option, then no remarks at all are used. The command:

fhist prog.c −u −r

would create a new revision of *prog.c* without asking for or saving any remarks about the edit.

−Remark_String *text*

It is also possible to specify the remarks directly on the command line. You may only use this option once.

−Extract [*edit*]

To retrieve a previous revision of a module, you specify the name of the module and use the **−Extract** option to specify the edit number you want retrieved. Edit numbers are assigned sequentially starting with 1. Thus the initial version of the module has edit number 1, the first revision has edit number 2, and so on until the latest revision. If the **−Extract** option is not used, or if no edit number is supplied for it, then the latest edit number is extracted. Therefore, this is the default action if no options at all are specified.

Edit numbers can also be zero, negative, or be a name with an optional offset. The number zero represents the latest edit number, and negative numbers indicate edit numbers backwards from the latest edit number. Edit names represent edit numbers whose name had been set by using the **−Name** option. For example, if edit number 10 was associated with the name *foo*, then the edit name *foo* represents 10, *foo-4* represents edit number6, and *foo+2* represents edit number 12. The special reserved names *oldest* and *newest* refer to the oldest and newest versions of the module in the edit history.

As an example of retrievals, assume that you have saved ten versions of the module *prog.c*. The following commands will then extract the versions of the file with the specified edit numbers:

fhist prog.c
version 10 (the latest)

fhist prog.c −e 9
version 9 (the version just prior)

fhist prog.c −e oldest
version 1 (the oldest version)

fhist prog.c −e −2
version 8 (latest version − 2)

The output filename is again defaulted to the module name. So when the module *prog.c* is extracted, the specified version of the module is written to the *prog.c* file.

In order to prevent accidental overwriting of a file, the *fhist* program will by default ask you if overwriting is permitted if that would occur. A common mistake is to edit *prog.c*, and then try to update the module, but forget to specify the **−u** option. Then the *fhist* program would try to extract the newest version of the module, and thus overwrite the file with the new changes. Asking the question allows you to notice your mistake, and prevent the overwriting.

−Output *filename*

You can change the output filename using the **−Output** option. Thus, the command:

fhist prog.c −o newprog.c

will extract the latest version of the module *prog.c*, and put it into the file *newprog.c*. Once again, the file "prog.c" is ignored, whether or not it existed.

–Force_Write

This option will force overwriting of the file, thus never asking you if overwriting is permitted. This is often useful in shell scripts, or when you are *sure* that you want to overwrite any existing file.

–No_Write

This option is the no-overwrite option, and will cause any existing files to *not* be overwritten, again without asking you. This is useful if you already have some of the modules in your directory, and you want to extract the rest of the modules without overwriting the ones you already have. Specifying both **–Fore_Write** and **–No_Write** is an error.

–Terminal [*edit*]

This option is used to output an extracted module to the standard output, instead of writing it to a file. This is useful in order to view the beginning of a version of the file. This can be interrupted if you do not want to see the whole file.

–Modify *number*

When extracting a file, the *fhist* program looks for and updates special character sequences in the first few lines of the file. These special sequences are used for documentation purposes, such as describing the edit number the file is from. For speed of extraction and updating, these sequences are usually limited to the first 25 lines of the file, since the *fhist* program then does not have to examine the entire file. The **–Modify** option can be used to change the number of lines to be modified from the default value of 25. Specifying zero totally disables the special character sequences, whereas specifying a very large number will cause the sequences to be checked for each line of the file (and thus slow the *fhist* program down).

Each special sequence is of the form **[# keyword value, keyword value, ..., keyword value #] ,** where each *keyword* describes an item, and each *value* is the value for the preceding keyword. The keywords can be in upper or lower case, or both. The single space following the **[#**, following each comma, and preceding the **#]** must be present. If the sequence is wrong, an unknown keyword is used, the line is longer than 200 characters, or more than four keywords are used, then the whole line will not be changed. The current keywords which can be used are the following:

edit	The edit number
date	The date that the edit was created
user	The user name of the user who created the edit
module	The module name

In order to use this special character sequence, you simply insert it into your module inside of a comment (within the first few lines). When this is done, the value parts of the sequence can be null. For example, if you want to put a special sequence into a program called *delete.c*, then you could edit the first few lines as follows:

```
/*
 * Delete – program to delete files
 * [# Edit, Date #]
 */
```

When an extract is done, the proper edit number and date are automatically inserted as the new values. Thus, if you extract edit 23 of the module *delete.c* which had been created on 8 August 89, then the resulting file would begin:

```
/*
 * Delete – program to delete files
 * [# Edit 23, Date 8-Aug-89 #]
 */
```

When updating a module, it is never necessary to edit these sequences, as any old values will be removed and replaced with the new ones. Also, when using the **–d** or **–du** options (described below), lines with these sequences compare as if the values were null, and thus will not cause

spurious differences.

During an update, the special character sequences are read and any edit value found is compared against the current edit number of the module. If they differ, then the update fails. This provides an interlock check for the case of two users extracting the same version of a file, editing it, and then both updating it without knowledge of each other. In this case, the second user would fail, and then he can merge his edits with the previous user's edit and then retry the update. This checking is disabled if there is no special character sequence containing the edit keyword, the edit number value is null, or if the **–Forced_Update** option is used to indicate that the check is not needed.

–No_Keywords

This option may be used to disable the use of the keyword special character sequences described above. Text containing keyword sequences is treated as plain text. Note: the **–No_Keywords** option does *not* imply the **–BINary** option.

–Name *string*

This option is used to associate a name for the newest version of a module. It can be given along with the **–CReate**, **–Update**, or **–Difference_Update** options, to specify a name for the new version of the module. It can also be given by itself in order to specify a name for the newest version of a module. Each edit number can have many names associated with it, so this will not remove any previously defined name for the edit. This option is useful to correlate many modules together. For example, when a new version of a program is ready to be released, you could give each module of the program the same name *release1*. Then in the future, you can recreate the sources making up that release by extracting the edits with the name *release1* for every module. Edit names cannot begin with a digit, and cannot contain plus or minus signs. These rules prevent ambiguous parsing of edit numbers for the **–Extract**, **–Terminal**, **–ALL**, and **–List** options.

–List [*edit1* [*edit2*]]

This option prints a list of edits for the module, giving the user name, date, user remarks, and names specified for the edits. If no edit number is supplied, then all edits are printed in reverse order. If a single edit number is supplied, then only that edit number is printed. If two edit numbers are supplied, then all edits in the specified range are printed. The output from this option defaults to the terminal. You can use the **–Output** option to save the results to a file.

–Difference [*edit1* [*edit2*]]

This option is used to display the differences between two versions of a module, or a file and a version of a module. There are three modes for this action, depending on how many edit numbers are supplied. These modes are illustrated by the following examples:

fhist foo.c –d
: Compare latest version against file "foo.c"

fhist foo.c –d 3
: Compare version 3 against file "foo.c"

fhist foo.c –d 3 4
: Compare version 3 against version 4

This option accepts the **–Input** option to specify the file to be compared. When using the **–Difference** option, the output defaults to the terminal. Therefore, you must use **–Output** if you wish the differences saved to a file. Using **–Quick** with **–Difference** will only output a quick summary of the changes, instead of the detailed changes. This summary only supplies the number of lines inserted, deleted, and unchanged between the files. Using **–What** with **–Difference** will display all of both files, showing in detail what the differences are using change bars.

The **–Difference** option may need to write one or two temporary files in order to extract old versions of a module to be compared. These files have names like *T$n_nnn* . They are deleted again just before differences are output, so that stopping the output before it is complete will not leave these files around. The temporary files are usually written to the current directory. If this is not

reasonable because of permission or quota problems, then you can specify the directory for writing the temporary files into. This is done by defining the *TMPDIR* environment variable to be the path of the directory.

−Difference_Update

This option combines the effects of the **−Difference** and **−Update** options. It displays the differences between a file and the latest version of a module. If there are any differences, it then proceeds to perform an update of the module with that file, asking for remarks as usual. This option is very useful when used with wildcarded module names. Then you can update just those modules which were changed by an edit session, and see the changes for each module before typing the appropriate remark for each module.

You may specify both of the **−Difference** and **−Update** options, or you may use this option. The results are identical.

−Conditional_Update

This option conditionally updates a module. That is, it will only do an update if there are any differences between a file and the latest version of a module. This is convenient when related changes are made to many modules in a directory, and one command using wildcards can update just those modules that were changed.

The **−CReate** option may be combined with the **−Update** or **−Conditional_Update** options to create the file if required.

−CLean

This option is used to remove files which match the newest versions of modules. If a file exists which matches the newest version of a module, then the file is deleted, otherwise it is kept. This option is used to clean up a work directory after building a new version of a product. This option is especially useful when used with the **−ALL** option. It will also accept the **−Input** option to specify a directory containing the files to be cleaned.

−CHeck

This option is used to find out if a file does not match the latest version of a module. If so, a message is given. If the file does match, no output occurs. This option is thus useful to determine which files have been modified and in need of updating. The **−ALL** option is defaulted for this option, since it is usually used for all modules. For example,

fhist −CHeck

will report on all files which are different than the latest modules. If **−Quick** is specified, then the output will consist of the module names with no other output. This is useful for the backquote operator in shell scripts for referencing the modules which are out of date. The **−CHeck** option will also accept the **−Input** option.

−PRune *edit*

This option is used to permanently remove early edits from an edit history. This is useful if you wish to cut down on the amount of disk space taken by an edit history file, or when you want to start another release of a file, and want a copy of the edit history file for that new release. The option takes an edit number to preserve, and all edits in the edit history file before that edit are deleted, and can no longer be referenced. For example, to keep only the current edit plus the previous 10 edits of the module *file*, you could use the command:

fhist file −prune −10

Since the **−PRune** option is unrecoverable (unless backup files are available), the *fhist* program asks the user to verify that the prune is really wanted. The **−Forced_Update** option can be used to bypass this verification.

−ALL

This option can be used with any of the action options. It means perform the operation for all modules in the module storage directory. Alternatively, you can specify multiple module names on the command line, and the actions will be performed with those modules. You cannot specify both **−ALL** and module names.

When using multiple modules or the **−ALL** option, the **−Input** and **−Output** options have a slightly different meaning. In these cases, the **−Input** and **−Output** arguments are a directory name which contains filenames with the same name as the module names. If the argument is not a directory, then an error is given. This feature is useful for example, to extract all the modules and place them into some remote directory, as in:

fhist −all −e −o tempdir

You should be careful when specifying numeric edit numbers for multiple modules. Most probably, a particular edit number is not appropriate for multiple modules, since changes corresponding to a particular edit number are not usually related. Using named edits avoids these problems. As an example, if you wanted to extract every module which had an edit that was named *rev3*, then you could use the command:

fhist −all −e rev3

Some other useful examples of commands which use multiple modules are:

fhist *.c −create
fhist −check −all
fhist −cu −all

−Verbose

This option can be specified with any other action, and outputs status information about the progress of the action. This is useful for debugging of problems, or just for amusement when the system is slow or a large file is being processed. It accepts a numeric argument to indicate the verbosity for output. The levels are as follows:

0 No output at all (except for errors).

1 Single-line output describing action (default).

2 Detailed status as action proceeds.

−Help

Give some help on how to use the *fhist* program.

−VERSion

Show what version of *fhist* is running.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "−help, "−HELP" and "−h" are all interpreted to mean the **−Help** option. The argument "−hlp" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line.

The GNU long option names are understood. Since all option names for *fhist* are long, this means ignoring the extra leading '−'. The "**−***option***=***value*" convention is also understood.

FILE NAME EXPANSION

As a convenience, if a pathname begins with a period and a environment variable exists with that name, then the value of the environment variable will be used as the actual pathname. For example, if a environment variable of *.FOO* has the value *this.is.a.long.name*, then the command

fhist −o .FOO

is actually equivilant to the command

fhist −o this.is.a.long.name

If you want to prevent the expansion of a pathname which begins with a period, then you can use an alternate form for the pathname, as in:

fhist −o ./.FOO

BINARY FILES

In general, fhist can handle all text files you throw at it, even international text with unusual encodings. However, fhist is *unable* to cope elegantly with files which contain the NUL character.

The *fcomp*(1) program simply prints a warning, and continues, you need to know that it converts NUL characters into an 0x80 value before performing the comparison.

The *fmerge*(1) program also converts the NUL character to an 0x80 value before merging, after a warning, and any output file will contain this value, rather than the original NUL character.

The *fhist*(1) program, however, generates a fatal error if any input file contains NUL characters. This is intended to protect your source files for unintentional corruption. Use **−BINary** for files which absolutely must contain NUL characters.

EXIT STATUS

The *fhist* program will exit with a status of 1 on any error. The *fhist* program will only exit with a status of 0 if there are no errors.

REFERENCES

This program is based on the algorithm in

An O(ND) Difference Algorithm and Its Variations, Eugene W. Myers, TR 85-6, 10-April-1985, Department of Computer Science, The University of Arizona, Tuscon, Arizona 85721.

See also:

A File Comparison Program, Webb Miller and Eugene W. Myers, Software Practice and Experience, Volume 15, No. 11, November 1985.

COPYRIGHT

fhist version 1.18.D001
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is derived from a work
Copyright © 1990 David I. Bell.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

AUTHORS

Peter Miller Web: http://miller.emu.id.au/pmiller/
/\/* E-Mail: pmiller@opensource.org.au

David I. Bell Web: http://www.canb.auug.org.au/~dbell
E-Mail: dbell@canb.auug.org.au

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program – to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product,

and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license,

or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s

essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section

13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and a brief idea of what it does.*
> Copyright (C) *year name of author*
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program 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.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author>
> This program comes with ABSOLUTELY NO WARRANTY; for details type "show w". This is free software, and you are welcome to redistribute it under certain conditions; type "show c" for details.

The hypothetical commands "show w" and "show c" should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

NAME

fmerge – merge files

SYNOPSIS

fmerge [*option...*] *basefile fileA fileB*

fmerge –Help

fmerge –VERSion

DESCRIPTION

The *fmerge* program is used to compare the changes between two different descendants of a base file, and creates an output file which contains both sets of changes. This is useful when two users both take the same version of a file and make independent edits to it, and then later want to create a file which contains both sets of edits. In such a use, the original file that both sets of edits is derived from is called the *base file*. The two files containing the edits are called *file A* and *file B*.

The command:

fmerge basefile fileA fileB –o outputfile

produces the output file which contains the edits contained in *file A* and *file B*, based on the *base file* . If the **–Output** option was not used, or if no outputfile is specified, then the merged lines are typed to the standard output. The order of specifying *file A and file B* is usually unimportant.

The *fmerge* program can also be used to remove earlier edits made to a module. To do this, make the version containing the edits you want to delete be the basefile. Make the version previous to the edit you want deleted be file A. Finally, make the most recent version of the file which contains the other edits (including the one you want deleted) be file B. Then the result of merging will be the newest version of the module minus the changes made by the edit you wanted removed. For example, if three successive versions of some module have the names *edit10*, *edit11* and *edit12*, and you want the changes done by *edit11* to be undone, but still want the changes done by *edit12*, then you use the command:

fmerge edit11 edit10 edit12 –o outputfile

While merging the two sets of edits, fmerge may discover conflicts. A conflict occurs when the same line of the base file is changed by both of the two sets of edits. The change can be due to new lines being inserted, lines being deleted, or both. When conflicts occur, the output file contains conflict identification lines, which are lines containing the string *'/–/–/–/'*. These lines indicate the region where the two sets of edits are incompatible. You must then edit the output file and remove these lines, and in addition correct the conflicts manually in order to produce the correct result.

OPTIONS

The following options are understood:

–Conflicts [*conflictfile*]

Since conflicts due to deletions are invisible in the output file, and inserts do not specify which of the two edits inserted the lines, there is an alternative output format from the *fmerge* program. This output format describes what happens to each line of the base file, so that conflicts are easier to detect and fix. The command:

fmerge basefile fileA fileB –c conflictfile

produces the file describing the results of the merge in detail. If the **–Conflicts** option is specified without any conflictfile name, then the conflicts are send to the standard output.

If there are conflicts, and the **–Conflicts** options is not specified, the *fmerge* program will exit with a status of 1.

The conflict file contains lines which contain three characters and then some text. The first three characters describe what is happening to the base file at that point. These characters are the following:

IA This line was inserted by file A.

DA This line was deleted by file A.

IB This line was inserted by file B.

DB This line was deleted by file B.

<blanks>
This line is unchanged.

X This is a conflict identification line.

U There are unspecified unchanged lines here.

Each set of conflicts is flagged by three identification lines. The first line indicates the beginning of the conflict, and specifies the line numbers for the base file and two divergent files. The second conflict identification line separates lines changed by file A from lines changed by file B. The third conflict identification indicates the end of the conflict.

You can edit this conflict file to remove the conflicts. This involves deleting the conflict identification lines, and changing the conflicting lines as necessary to fix the conflict. While doing this, remember to leave three blank characters at the front of any new lines you insert while correcting the conflicts. When you are done, there should be no lines which begin with an 'X' in the file. All other lines can remain. Then you can use the command:

fmerge conflictfile −o outputfile

to create the new output file which has the desired data. Once again, if no −*Output* option or outputfile is used, the output is send to the standard output.

−Unchanged *number*

Besides physical conflicts, there can be logical conflicts. These are changes made to different lines in the base file such that the program is no longer correct. Such conflicts cannot be detected by a program, and so these must be checked manually. In order to make this process easier, the **−Unchanged** option can be used to reduce the size of the conflict file to only include regions near changed lines. This file can then be examined in order to detect possible logical conflicts. As an example, the command:

fmerge basefile fileA fileB −c −u 3

will send to the standard output all changes made by either sets of edits, with only three unchanged lines surrounding each edit.

When using the **−Unchanged** option, the conflict file will contain lines starting with 'U'. These represent unchanged lines, and the number following the letter is the number of unchanged lines. The resulting conflict file cannot be read to produce an output file because of the missing lines. If this is attempted, an error will be generated.

It is possible to use both **−Output** and **−Conflicts** in the same command. Thus you can produce the output file which you hope is correct, and also produce the conflict file which you can use to check for logical conflicts.

−Verbose [*number*]

This option can be specified with any other action, and outputs status information about the progress of the action. This is useful for debugging of problems, or just for amusement when the system is slow or a large file is being processed. It accepts a numeric argument to indicate the verbosity for output. The levels are as follows:

0 No output at all (except for errors).

1 Single-line output describing action (default).

2 Detailed status as action proceeds.

−Failures *number*

This option restricts the number of physical conflicts that are allowed before failing. This is used if you are not interested in the results if there are too many conflicts.

−Help
Give some help on how to use the *fmerge* program.

−Ignore
Ignore all conflicts.

−Ignore_Identical_Conflicts
The option may be used to suppress conflicts which make identical deletes, or identical inserts, or identical changes. This is often desirable when merging two source code branches.

−VERSion
Show what version of *fmerge* is running.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "−help, "−HELP" and "−h" are all interpreted to mean the **−Help** option. The argument "−hlp" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line.

The GNU long option names are understood. Since all option names for *fmerge* are long, this means ignoring the extra leading '−'. The "−*option*=*value*" convention is also understood.

FILE NAME EXPANSION

As a convenience, if a pathname begins with a period and a environment variable exists with that name, then the value of the environment variable will be used as the actual pathname. For example, if a environment variable of *.FOO* has the value *this.is.a.long.name*, then the command

fmerge −o .FOO

is actually equivilant to the command

fmerge −o this.is.a.long.name

If you want to prevent the expansion of a pathname which begins with a period, then you can use an alternate form for the pathname, as in:

fmerge −o ./.FOO

BINARY FILES

In general, fmerge can handle all text files you throw at it, even international text with unusual encodings. However, fmerge is *unable* to cope elegantly with files which contain the NUL character.

The *fcomp*(1) program simply prints a warning, and continues, you need to know that it converts NUL characters into an 0x80 value before performing the comparison.

The *fmerge*(1) program also converts the NUL character to an 0x80 value before merging, after a warning, and any output file will contain this value, rather than the original NUL character.

The *fhist*(1) program, however, generates a fatal error if any input file contains NUL characters. This is intended to protect your source files for unintentional corruption. Use **−BINary** for files which absolutely must contain NUL characters.

EXIT STATUS

The *fmerge* program will exit with a status of 1 on any error. The *fmerge* program will only exit with a status of 0 if there are no errors.

REFERENCES

This program is based on the algorithm in

An O(ND) Difference Algorithm and Its Variations, Eugene W. Myers, TR 85-6, 10-April-1985, Department of Computer Science, The University of Arizona, Tuscon, Arizona 85721.

See also:

A File Comparison Program, Webb Miller and Eugene W. Myers, Software Practice and Experience, Volume 15, No. 11, November 1985.

COPYRIGHT

fmerge version 1.18.D001
Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2009 Peter Miller;

This program is derived from a work
Copyright © 1990 David I. Bell.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program 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.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

AUTHORS

Peter Miller Web: http://miller.emu.id.au/pmiller/
/\/* E-Mail: pmiller@opensource.org.au

David I. Bell Web: http://www.canb.auug.org.au/~dbell
E-Mail: dbell@canb.auug.org.au

	The README File	1
	Release Notes	3
	How to Build the Sources	6
	Internationalization	10
fcomp(1)	compare two files	12
fhist(1)	record file modification history	15
fhist_lic(1)	GNU General Public License	23
fmerge(1)	merge competing file edits	32

fcomp(1)	12	fcomp - file	compare
fcomp(1)	12		fcomp - file compare
fhist(1)	15		fhist - file history
fcomp(1)	12	fcomp -	file compare
fhist(1)	15	fhist -	file history
fmerge(1)	32	fmerge - merge	files
fmerge(1)	32		fmerge - merge files
fhist(1)	15	fhist - file	history
fcomp(1)	12	require_	index
fhist(1)	15	require_	index
fmerge(1)	32	require_	index
fmerge(1)	32	fmerge -	merge files
fcomp(1)	12		require_index
fhist(1)	15		require_index
fmerge(1)	32		require_index