NAME

patch − apply a diff file to an original

SYNOPSIS

patch [*options*] [*originalfile* [*patchfile*]]

but usually just

patch −p*num* <*patchfile*

DESCRIPTION

patch takes a patch file *patchfile* containing a difference listing produced by the **diff** program and applies those differences to one or more original files, producing patched versions. Normally the patched versions are put in place of the originals. Backups can be made; see the **−b** or **−−backup** option. The names of the files to be patched are usually taken from the patch file, but if there's just one file to be patched it can specified on the command line as *originalfile*.

Upon startup, patch attempts to determine the type of the diff listing, unless overruled by a **−c** (**−−context**), **−e** (**−−ed**), **−n** (**−−normal**), or **−u** (**−−unified**) option. Context diffs (old-style, new-style, and unified) and normal diffs are applied by the **patch** program itself, while **ed** diffs are simply fed to the **ed**(1) editor via a pipe.

patch tries to skip any leading garbage, apply the diff, and then skip any trailing garbage. Thus you could feed an article or message containing a diff listing to **patch**, and it should work. If the entire diff is indented by a consistent amount, or if a context diff contains lines ending in CRLF or is encapsulated one or more times by prepending "− " to lines starting with "−" as specified by Internet RFC 934, this is taken into account. After removing indenting or encapsulation, lines beginning with # are ignored, as they are considered to be comments.

With context diffs, and to a lesser extent with normal diffs, **patch** can detect when the line numbers mentioned in the patch are incorrect, and attempts to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, **patch** scans both forwards and backwards for a set of lines matching the context given in the hunk. First **patch** looks for a place where all lines of the context match. If no such place is found, and it's a context diff, and the maximum fuzz factor is set to 1 or more, then another scan takes place ignoring the first and last line of context. If that fails, and the maximum fuzz factor is set to 2 or more, the first two and last two lines of context are ignored, and another scan is made. (The default maximum fuzz factor is 2.) If **patch** cannot find a place to install that hunk of the patch, it puts the hunk out to a reject file, which normally is the name of the output file plus a **.rej** suffix, or # if **.rej** would generate a file name that is too long (if even appending the single character # makes the file name too long, then # replaces the file name's last character). (The rejected hunk comes out in ordinary context diff form regardless of the input patch's form. If the input was a normal diff, many of the contexts are simply null.) The line numbers on the hunks in the reject file may be different than in the patch file: they reflect the approximate location patch thinks the failed hunks belong in the new file rather than the old one.

As each hunk is completed, you are told if the hunk failed, and if so which line (in the new file) **patch** thought the hunk should go on. If the hunk is installed at a different line from the line number specified in the diff you are told the offset. A single large offset *may* indicate that a hunk was installed in the wrong place. You are also told if a fuzz factor was used to make the match, in which case you should also be slightly suspicious. If the **−−verbose** option is given, you are also told about hunks that match exactly.

If no original file *origfile* is specified on the command line, **patch** tries to figure out from the leading garbage what the name of the file to edit is, using the following rules.

First, **patch** takes an ordered list of candidate file names as follows:

- If the header is that of a context diff, **patch** takes the old and new file names in the header. A name is ignored if it does not have enough slashes to satisfy the **−p***num* or **−−strip=***num* option. The name **/dev/null** is also ignored.
- If there is an **Index:** line in the leading garbage and if either the old and new names are both absent or if **patch** is conforming to POSIX, **patch** takes the name in the **Index:** line.

- For the purpose of the following rules, the candidate file names are considered to be in the order (old, new, index), regardless of the order that they appear in the header.

Then **patch** selects a file name from the candidate list as follows:

- If some of the named files exist, **patch** selects the first name if conforming to POSIX, and the best name otherwise.
- If **patch** is not ignoring RCS, ClearCase, Perforce, and SCCS (see the **−g** *num* or **−−get=***num* option), and no named files exist but an RCS, ClearCase, Perforce, or SCCS master is found, **patch** selects the first named file with an RCS, ClearCase, Perforce, or SCCS master.
- If no named files exist, no RCS, ClearCase, Perforce, or SCCS master was found, some names are given, **patch** is not conforming to POSIX, and the patch appears to create a file, **patch** selects the best name requiring the creation of the fewest directories.
- If no file name results from the above heuristics, you are asked for the name of the file to patch, and **patch** selects that name.

To determine the *best* of a nonempty list of file names, **patch** first takes all the names with the fewest path name components; of those, it then takes all the names with the shortest basename; of those, it then takes all the shortest names; finally, it takes the first remaining name.

Additionally, if the leading garbage contains a **Prereq:** line, **patch** takes the first word from the prerequisites line (normally a version number) and checks the original file to see if that word can be found. If not, **patch** asks for confirmation before proceeding.

The upshot of all this is that you should be able to say, while in a news interface, something like the following:

| patch −d /usr/src/local/blurfl

and patch a file in the **blurfl** directory directly from the article containing the patch.

If the patch file contains more than one patch, **patch** tries to apply each of them as if they came from separate patch files. This means, among other things, that it is assumed that the name of the file to patch must be determined for each diff listing, and that the garbage before each diff listing contains interesting things such as file names and revision level, as mentioned previously.

OPTIONS

−b or **−−backup**
Make backup files. That is, when patching a file, rename or copy the original instead of removing it. When backing up a file that does not exist, an empty, unreadable backup file is created as a placeholder to represent the nonexistent file. See the **−V** or **−−version−control** option for details about how backup file names are determined.

−−backup−if−mismatch
Back up a file if the patch does not match the file exactly and if backups are not otherwise requested. This is the default unless **patch** is conforming to POSIX.

−−no−backup−if−mismatch
Do not back up a file if the patch does not match the file exactly and if backups are not otherwise requested. This is the default if **patch** is conforming to POSIX.

−B *pref* or **−−prefix=***pref*
Prefix *pref* to a file name when generating its simple backup file name. For example, with **−B /junk/** the simple backup file name for **src/patch/util.c** is **/junk/src/patch/util.c**.

−−binary
Read and write all files in binary mode, except for standard output and **/dev/tty**. This option has no effect on POSIX-conforming systems. On systems like DOS where this option makes a difference, the patch should be generated by **diff −a −−binary**.

−c or **−−context**
Interpret the patch file as a ordinary context diff.

−d *dir* or **−−directory=***dir*
Change to the directory *dir* immediately, before doing anything else.

−D *define* or **−−ifdef=***define*
Use the **#ifdef** . . . **#endif** construct to mark changes, with *define* as the differentiating symbol.

−−dry−run
Print the results of applying the patches without actually changing any files.

−e or **−−ed**
Interpret the patch file as an **ed** script.

−E or **−−remove−empty−files**
Remove output files that are empty after the patches have been applied. Normally this option is unnecessary, since **patch** can examine the time stamps on the header to determine whether a file should exist after patching. However, if the input is not a context diff or if **patch** is conforming to POSIX, **patch** does not remove empty patched files unless this option is given. When **patch** removes a file, it also attempts to remove any empty ancestor directories.

−f or **−−force**
Assume that the user knows exactly what he or she is doing, and do not ask any questions. Skip patches whose headers do not say which file is to be patched; patch files even though they have the wrong version for the **Prereq:** line in the patch; and assume that patches are not reversed even if they look like they are. This option does not suppress commentary; use **−s** for that.

−F *num* or **−−fuzz=***num*
Set the maximum fuzz factor. This option only applies to diffs that have context, and causes **patch** to ignore up to that many lines in looking for places to install a hunk. Note that a larger fuzz factor increases the odds of a faulty patch. The default fuzz factor is 2, and it may not be set to more than the number of lines of context in the context diff, ordinarily 3.

−g *num* or **−−get=***num*
This option controls **patch**'s actions when a file is under RCS or SCCS control, and does not exist or is read-only and matches the default version, or when a file is under ClearCase or Perforce control and does not exist. If *num* is positive, **patch** gets (or checks out) the file from the revision control system; if zero, **patch** ignores RCS, ClearCase, Perforce, and SCCS and does not get the file; and if negative, **patch** asks the user whether to get the file. The default value of this option is given by the value of the **PATCH_GET** environment variable if it is set; if not, the default value is zero if **patch** is conforming to POSIX, negative otherwise.

−−help
Print a summary of options and exit.

−i *patchfile* or **−−input=***patchfile*
Read the patch from *patchfile*. If *patchfile* is **−**, read from standard input, the default.

−l or **−−ignore−whitespace**
Match patterns loosely, in case tabs or spaces have been munged in your files. Any sequence of one or more blanks in the patch file matches any sequence in the original file, and sequences of blanks at the ends of lines are ignored. Normal characters must still match exactly. Each line of the context must still match a line in the original file.

−n or **−−normal**
Interpret the patch file as a normal diff.

−N or **−−forward**
Ignore patches that seem to be reversed or already applied. See also **−R**.

−o *outfile* or **−−output=***outfile*
Send output to *outfile* instead of patching files in place. Do not use this option if *outfile* is one of the files to be patched.

−p*num* or **−−strip=***num*
Strip the smallest prefix containing *num* leading slashes from each file name found in the patch file. A sequence of one or more adjacent slashes is counted as a single slash. This controls how file names found in the patch file are treated, in case you keep your files in a different directory than the person who sent out the patch. For example, supposing the file name in the patch file was

/u/howard/src/blurfl/blurfl.c

setting **−p0** gives the entire file name unmodified, **−p1** gives

u/howard/src/blurfl/blurfl.c

without the leading slash, **−p4** gives

blurfl/blurfl.c

and not specifying **−p** at all just gives you **blurfl.c**. Whatever you end up with is looked for either in the current directory, or the directory specified by the **−d** option.

−−posix
Conform more strictly to the POSIX standard, as follows.

- Take the first existing file from the list (old, new, index) when intuiting file names from diff headers.
- Do not remove files that are empty after patching.
- Do not ask whether to get files from RCS, ClearCase, Perforce, or SCCS.
- Require that all options precede the files in the command line.
- Do not backup files when there is a mismatch.

−−quoting−style=*word*
Use style *word* to quote output names. The *word* should be one of the following:

literal Output names as-is.

shell Quote names for the shell if they contain shell metacharacters or would cause ambiguous output.

shell-always
Quote names for the shell, even if they would normally not require quoting.

c Quote names as for a C language string.

escape Quote as with **c** except omit the surrounding double-quote characters.

You can specify the default value of the **−−quoting−style** option with the environment variable **QUOTING_STYLE**. If that environment variable is not set, the default value is **shell**.

−r *rejectfile* or **−−reject−file=***rejectfile*
Put rejects into *rejectfile* instead of the default **.rej** file.

−R or **−−reverse**
Assume that this patch was created with the old and new files swapped. (Yes, I'm afraid that does happen occasionally, human nature being what it is.) **patch** attempts to swap each hunk around before applying it. Rejects come out in the swapped format. The **−R** option does not work with **ed** diff scripts because there is too little information to reconstruct the reverse operation.

If the first hunk of a patch fails, **patch** reverses the hunk to see if it can be applied that way. If it can, you are asked if you want to have the **−R** option set. If it can't, the patch continues to be applied normally. (Note: this method cannot detect a reversed patch if it is a normal diff and if the first command is an append (i.e. it should have been a delete) since appends always succeed, due to the fact that a null context matches anywhere. Luckily, most patches add or change lines rather than delete them, so most reversed normal diffs begin with a delete, which fails, triggering the heuristic.)

−s or **−−silent** or **−−quiet**
Work silently, unless an error occurs.

−t or **−−batch**
Suppress questions like **−f**, but make some different assumptions: skip patches whose headers do not contain file names (the same as **−f**); skip patches for which the file has the wrong version for the **Prereq:** line in the patch; and assume that patches are reversed if they look like they are.

−T or **−−set−time**
Set the modification and access times of patched files from time stamps given in context diff headers, assuming that the context diff headers use local time. This option is not recommended, because patches using local time cannot easily be used by people in other time zones, and because local time stamps are ambiguous when local clocks move backwards during daylight-saving time adjustments. Instead of using this option, generate patches with UTC and use the **−Z** or **−−set−utc** option

instead.

−u or **−−unified**
Interpret the patch file as a unified context diff.

−v or **−−version**
Print out **patch**'s revision header and patch level, and exit.

−V *method* or **−−version−control=***method*
Use *method* to determine backup file names. The method can also be given by the **PATCH_VERSION_CONTROL** (or, if that's not set, the **VERSION_CONTROL**) environment variable, which is overridden by this option. The method does not affect whether backup files are made; it affects only the names of any backup files that are made.

The value of *method* is like the GNU Emacs 'version-control' variable; **patch** also recognizes synonyms that are more descriptive. The valid values for *method* are (unique abbreviations are accepted):

existing or **nil**
Make numbered backups of files that already have them, otherwise simple backups. This is the default.

numbered or **t**
Make numbered backups. The numbered backup file name for *F* is *F*.~*N*~ where *N* is the version number.

simple or **never**
Make simple backups. The **−B** or **−−prefix**, **−Y** or **−−basename−prefix**, and **−z** or **−−suffix** options specify the simple backup file name. If none of these options are given, then a simple backup suffix is used; it is the value of the **SIMPLE_BACKUP_SUFFIX** environment variable if set, and is **.orig** otherwise.

With numbered or simple backups, if the backup file name is too long, the backup suffix ~ is used instead; if even appending ~ would make the name too long, then ~ replaces the last character of the file name.

−−verbose
Output extra information about the work being done.

−x *num* or **−−debug=***num*
Set internal debugging flags of interest only to **patch** patchers.

−Y *pref* or **−−basename−prefix=***pref*
Prefix *pref* to the basename of a file name when generating its simple backup file name. For example, with **−Y .del/** the simple backup file name for **src/patch/util.c** is **src/patch/.del/util.c**.

−z *suffix* or **−−suffix=***suffix*
Use *suffix* as the simple backup suffix. For example, with **−z -** the simple backup file name for **src/patch/util.c** is **src/patch/util.c-**. The backup suffix may also be specified by the **SIMPLE_BACKUP_SUFFIX** environment variable, which is overridden by this option.

−Z or **−−set−utc**
Set the modification and access times of patched files from time stamps given in context diff headers, assuming that the context diff headers use Coordinated Universal Time (UTC, often known as GMT). Also see the **−T** or **−−set−time** option.

The **−Z** or **−−set−utc** and **−T** or **−−set−time** options normally refrain from setting a file's time if the file's original time does not match the time given in the patch header, or if its contents do not match the patch exactly. However, if the **−f** or **−−force** option is given, the file time is set regardless.

Due to the limitations of **diff** output format, these options cannot update the times of files whose contents have not changed. Also, if you use these options, you should remove (e.g. with **make clean**) all files that depend on the patched files, so that later invocations of **make** do not get confused by the patched files' times.

ENVIRONMENT

PATCH_GET
This specifies whether **patch** gets missing or read-only files from RCS, ClearCase, Perforce, or SCCS by default; see the **−g** or **−−get** option.

POSIXLY_CORRECT
If set, **patch** conforms more strictly to the POSIX standard by default: see the **−−posix** option.

QUOTING_STYLE
Default value of the **−−quoting−style** option.

SIMPLE_BACKUP_SUFFIX
Extension to use for simple backup file names instead of **.orig**.

TMPDIR, **TMP**, **TEMP**
Directory to put temporary files in; **patch** uses the first environment variable in this list that is set. If none are set, the default is system-dependent; it is normally **/tmp** on Unix hosts.

VERSION_CONTROL or **PATCH_VERSION_CONTROL**
Selects version control style; see the **−v** or **−−version−control** option.

FILES

$TMPDIR/**p**∗
temporary files

/dev/tty
controlling terminal; used to get answers to questions asked of the user

SEE ALSO

diff(1), **ed**(1)

Marshall T. Rose and Einar A. Stefferud, Proposed Standard for Message Encapsulation, Internet RFC 934 <URL:ftp://ftp.isi.edu/in-notes/rfc934.txt> (1985-01).

NOTES FOR PATCH SENDERS

There are several things you should bear in mind if you are going to be sending out patches.

Create your patch systematically. A good method is the command **diff −Naur** *old new* where *old* and *new* identify the old and new directories. The names *old* and *new* should not contain any slashes. The **diff** command's headers should have dates and times in Universal Time using traditional Unix format, so that patch recipients can use the **−Z** or **−−set−utc** option. Here is an example command, using Bourne shell syntax:

LC_ALL=C TZ=UTC0 diff −Naur gcc−2.7 gcc−2.8

Tell your recipients how to apply the patch by telling them which directory to **cd** to, and which **patch** options to use. The option string **−Np1** is recommended. Test your procedure by pretending to be a recipient and applying your patch to a copy of the original files.

You can save people a lot of grief by keeping a **patchlevel.h** file which is patched to increment the patch level as the first diff in the patch file you send out. If you put a **Prereq:** line in with the patch, it won't let them apply patches out of order without some warning.

You can create a file by sending out a diff that compares **/dev/null** or an empty file dated the Epoch (1970-01-01 00:00:00 UTC) to the file you want to create. This only works if the file you want to create doesn't exist already in the target directory. Conversely, you can remove a file by sending out a context diff that compares the file to be deleted with an empty file dated the Epoch. The file will be removed unless **patch** is conforming to POSIX and the **−E** or **−−remove−empty−files** option is not given. An easy way to generate patches that create and remove files is to use GNU **diff**'s **−N** or **−−new−file** option.

If the recipient is supposed to use the **−p***N* option, do not send output that looks like this:

diff −Naur v2.0.29/prog/README prog/README
−−− v2.0.29/prog/README Mon Mar 10 15:13:12 1997
+++ prog/README Mon Mar 17 14:58:22 1997

because the two file names have different numbers of slashes, and different versions of **patch** interpret the file names differently. To avoid confusion, send output that looks like this instead:

diff −Naur v2.0.29/prog/README v2.0.30/prog/README
−−− v2.0.29/prog/README Mon Mar 10 15:13:12 1997
+++ v2.0.30/prog/README Mon Mar 17 14:58:22 1997

Avoid sending patches that compare backup file names like **README.orig**, since this might confuse **patch** into patching a backup file instead of the real file. Instead, send patches that compare the same base file names in different directories, e.g. **old/README** and **new/README**.

Take care not to send out reversed patches, since it makes people wonder whether they already applied the patch.

Try not to have your patch modify derived files (e.g. the file **configure** where there is a line **configure: configure.in** in your makefile), since the recipient should be able to regenerate the derived files anyway. If you must send diffs of derived files, generate the diffs using UTC, have the recipients apply the patch with the **−Z** or **−−set−utc** option, and have them remove any unpatched files that depend on patched files (e.g. with **make clean**).

While you may be able to get away with putting 582 diff listings into one file, it may be wiser to group related patches into separate files in case something goes haywire.

DIAGNOSTICS

Diagnostics generally indicate that **patch** couldn't parse your patch file.

If the **−−verbose** option is given, the message **Hmm...** indicates that there is unprocessed text in the patch file and that **patch** is attempting to intuit whether there is a patch in that text and, if so, what kind of patch it is.

patch's exit status is 0 if all hunks are applied successfully, 1 if some hunks cannot be applied, and 2 if there is more serious trouble. When applying a set of patches in a loop it behooves you to check this exit status so you don't apply a later patch to a partially patched file.

CAVEATS

Context diffs cannot reliably represent the creation or deletion of empty files, empty directories, or special files such as symbolic links. Nor can they represent changes to file metadata like ownership, permissions, or whether one file is a hard link to another. If changes like these are also required, separate instructions (e.g. a shell script) to accomplish them should accompany the patch.

patch cannot tell if the line numbers are off in an **ed** script, and can detect bad line numbers in a normal diff only when it finds a change or deletion. A context diff using fuzz factor 3 may have the same problem. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Of course, compiling without errors is a pretty good indication that the patch worked, but not always.

patch usually produces the correct results, even when it has to do a lot of guessing. However, the results are guaranteed to be correct only when the patch is applied to exactly the same version of the file that the patch was generated from.

COMPATIBILITY ISSUES

The POSIX standard specifies behavior that differs from **patch**'s traditional behavior. You should be aware of these differences if you must interoperate with **patch** versions 2.1 and earlier, which do not conform to POSIX.

- In traditional **patch**, the **−p** option's operand was optional, and a bare **−p** was equivalent to **−p0.** The **−p** option now requires an operand, and **−p 0** is now equivalent to **−p0**. For maximum compatibility, use options like **−p0** and **−p1**.

  Also, traditional **patch** simply counted slashes when stripping path prefixes; **patch** now counts pathname components. That is, a sequence of one or more adjacent slashes now counts as a single slash. For maximum portability, avoid sending patches containing // in file names.

- In traditional **patch**, backups were enabled by default. This behavior is now enabled with the **−b** or **−−backup** option.

  Conversely, in POSIX **patch**, backups are never made, even when there is a mismatch. In GNU **patch**, this behavior is enabled with the **−−no−backup−if−mismatch** option, or by conforming to POSIX with the **−−posix** option or by setting the **POSIXLY_CORRECT** environment variable.

  The **−b** *suffix* option of traditional **patch** is equivalent to the **−b −z** *suffix* options of GNU **patch**.

- Traditional **patch** used a complicated (and incompletely documented) method to intuit the name of the file to be patched from the patch header. This method did not conform to POSIX, and had a few gotchas. Now **patch** uses a different, equally complicated (but better documented) method that is optionally POSIX-conforming; we hope it has fewer gotchas. The two methods are compatible if the file names in the context diff header and the **Index:** line are all identical after prefix-stripping. Your patch is normally compatible if each header's file names all contain the same number of slashes.
- When traditional **patch** asked the user a question, it sent the question to standard error and looked for an answer from the first file in the following list that was a terminal: standard error, standard output, **/dev/tty**, and standard input. Now **patch** sends questions to standard output and gets answers from **/dev/tty**. Defaults for some answers have been changed so that **patch** never goes into an infinite loop when using default answers.
- Traditional **patch** exited with a status value that counted the number of bad hunks, or with status 1 if there was real trouble. Now **patch** exits with status 1 if some hunks failed, or with 2 if there was real trouble.
- Limit yourself to the following options when sending instructions meant to be executed by anyone running GNU **patch**, traditional **patch**, or a **patch** that conforms to POSIX. Spaces are significant in the following list, and operands are required.

  −c
  −d *dir*
  −D *define*
  −e
  −l
  −n
  −N
  −o *outfile*
  −p*num*
  −R
  −r *rejectfile*

BUGS

Please report bugs via email to **<bug-patch@gnu.org>**.

patch could be smarter about partial matches, excessively deviant offsets and swapped code, but that would take an extra pass.

If code has been duplicated (for instance with **#ifdef OLDCODE** ... **#else ... #endif**), **patch** is incapable of patching both versions, and, if it works at all, will likely patch the wrong one, and tell you that it succeeded to boot.

If you apply a patch you've already applied, **patch** thinks it is a reversed patch, and offers to un-apply the patch. This could be construed as a feature.

COPYING

AUTHORS

Larry Wall wrote the original version of **patch**. Paul Eggert removed **patch**'s arbitrary limits; added support for binary files, setting file times, and deleting files; and made it conform better to POSIX. Other contributors include Wayne Davison, who added unidiff support, and David MacKenzie, who

added configuration and backup support.

NAME

patch − apply changes to files

SYNOPSIS

patch [-blNR][-c| -e| -n][-d *dir***][-D** *define***][-i** *patchfile***]**
[-o *outfile***][-p** *num***][-r** *rejectfile***][**_file_**]**

DESCRIPTION

The *patch* utility shall read a source (patch) file containing any of the three forms of difference (diff) listings produced by the *diff* utility (normal, context, or in the style of *ed*) and apply those differences to a file. By default, *patch* shall read from the standard input.

The *patch* utility shall attempt to determine the type of the *diff* listing, unless overruled by a **-c**, **-e**, or **-n** option.

If the patch file contains more than one patch, *patch* shall attempt to apply each of them as if they came from separate patch files. (In this case, the application shall ensure that the name of the patch file is determinable for each *diff* listing.)

OPTIONS

The *patch* utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-b Save a copy of the original contents of each modified file, before the differences are applied, in a file of the same name with the suffix **.orig** appended to it. If the file already exists, it shall be overwritten; if multiple patches are applied to the same file, the **.orig** file shall be written only for the first patch. When the **-o** *outfile* option is also specified, *file* **.orig** shall not be created but, if *outfile* already exists, *outfile* **.orig** shall be created.

-c Interpret the patch file as a context difference (the output of the utility *diff* when the **-c** or **-C** options are specified).

-d *dir* Change the current directory to *dir* before processing as described in the EXTENDED DESCRIPTION section.

-D *define*
Mark changes with one of the following C preprocessor constructs:

```
#ifdef define
...
#endif
```

```
#ifndef define
...
#endif
```

optionally combined with the C preprocessor construct **#else**. If the patched file is processed with the C preprocessor, where the macro *define* is defined, the output shall contain the changes from the patch file; otherwise, the output shall not contain the patches specified in the patch file.

-e Interpret the patch file as an *ed* script, rather than a *diff* script.

-i *patchfile*
Read the patch information from the file named by the pathname *patchfile*, rather than the standard input.

-l (The letter ell.) Cause any sequence of <blank>s in the difference script to match any sequence of <blank>s in the input file. Other characters shall be matched exactly.

-n Interpret the script as a normal difference.

-N Ignore patches where the differences have already been applied to the file; by default, already-applied patches shall be rejected.

-o *outfile*

Instead of modifying the files (specified by the *file* operand or the difference listings) directly, write a copy of the file referenced by each patch, with the appropriate differences applied, to *outfile*. Multiple patches for a single file shall be applied to the intermediate versions of the file created by any previous patches, and shall result in multiple, concatenated versions of the file being written to *outfile*.

-p *num*

For all pathnames in the patch file that indicate the names of files to be patched, delete *num* pathname components from the beginning of each pathname. If the pathname in the patch file is absolute, any leading slashes shall be considered the first component (that is, **-p 1** shall remove the leading slashes). Specifying **-p 0** shall cause the full pathname to be used. If **-p** is not specified, only the basename (the final pathname component) shall be used.

-R Reverse the sense of the patch script; that is, assume that the difference script was created from the new version to the old version. The **-R** option cannot be used with *ed* scripts. The *patch* utility shall attempt to reverse each portion of the script before applying it. Rejected differences shall be saved in swapped format. If this option is not specified, and until a portion of the patch file is successfully applied, *patch* attempts to apply each portion in its reversed sense as well as in its normal sense. If the attempt is successful, the user shall be prompted to determine whether the **-R** option should be set.

-r *rejectfile*

Override the default reject filename. In the default case, the reject file shall have the same name as the output file, with the suffix **.rej** appended to it; see Patch Application .

OPERANDS

The following operand shall be supported:

file A pathname of a file to patch.

STDIN

See the INPUT FILES section.

INPUT FILES

Input files shall be text files.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *patch*:

LANG Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of IEEE Std 1003.1-2001, Section 8.2, Internationalization Variables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL

If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH

Determine the location of message catalogs for the processing of *LC_MESSAGES* .

LC_TIME

Determine the locale for recognizing the format of file timestamps written by the *diff* utility in a context-difference input file.

ASYNCHRONOUS EVENTS

Default.

STDOUT

Not used.

STDERR

The standard error shall be used for diagnostic and informational messages.

OUTPUT FILES

The output of the *patch* utility, the save files (**.orig** suffixes), and the reject files (**.rej** suffixes) shall be text files.

EXTENDED DESCRIPTION

A patch file may contain patching instructions for more than one file; filenames shall be determined as specified in Filename Determination . When the **-b** option is specified, for each patched file, the original shall be saved in a file of the same name with the suffix **.orig** appended to it.

For each patched file, a reject file may also be created as noted in Patch Application . In the absence of a **-r** option, the name of this file shall be formed by appending the suffix **.rej** to the original filename.

Patch File Format

The patch file shall contain zero or more lines of header information followed by one or more patches. Each patch shall contain zero or more lines of filename identification in the format produced by *diff* **-c**, and one or more sets of *diff* output, which are customarily called *hunks*.

The *patch* utility shall recognize the following expression in the header information:

Index: *pathname*

The file to be patched is named *pathname*.

If all lines (including headers) within a patch begin with the same leading sequence of <blank>s, the *patch* utility shall remove this sequence before proceeding. Within each patch, if the type of difference is context, the *patch* utility shall recognize the following expressions:

*** *filename timestamp*

The patches arose from *filename*.

--- *filename timestamp*

The patches should be applied to *filename*.

Each hunk within a patch shall be the *diff* output to change a line range within the original file. The line numbers for successive hunks within a patch shall occur in ascending order.

Filename Determination

If no *file* operand is specified, *patch* shall perform the following steps to determine the filename to use:

1. If the type of *diff* is context, the *patch* utility shall delete pathname components (as specified by the **-p** option) from the filename on the line beginning with **"***"** , then test for the existence of this file relative to the current directory (or the directory specified with the **-d** option). If the file exists, the *patch* utility shall use this filename.

2. If the type of *diff* is context, the *patch* utility shall delete the pathname components (as specified by the **-p** option) from the filename on the line beginning with **"---"** , then test for the existence of this file relative to the current directory (or the directory specified with the **-d** option). If the file exists, the *patch* utility shall use this filename.

3. If the header information contains a line beginning with the string **Index:**, the *patch* utility shall delete pathname components (as specified by the **-p** option) from this line, then test for the existence of this file relative to the current directory (or the directory specified with the **-d** option). If the file exists, the *patch* utility shall use this filename.

4. If an **SCCS** directory exists in the current directory, *patch* shall attempt to perform a *get* **-e SCCS/s.** *filename* command to retrieve an editable version of the file. If the file exists, the *patch* utility shall use this filename.

5. The *patch* utility shall write a prompt to standard output and request a filename interactively from the controlling terminal (for example, **/dev/tty**).

Patch Application

If the **-c**, **-e**, or **-n** option is present, the *patch* utility shall interpret information within each hunk as a context difference, an *ed* difference, or a normal difference, respectively. In the absence of any of these options, the *patch* utility shall determine the type of difference based on the format of information within the hunk.

For each hunk, the *patch* utility shall begin to search for the place to apply the patch at the line number at the beginning of the hunk, plus or minus any offset used in applying the previous hunk. If lines matching the hunk context are not found, *patch* shall scan both forwards and backwards at least 1000 bytes for a set of lines that match the hunk context.

If no such place is found and it is a context difference, then another scan shall take place, ignoring the first and last line of context. If that fails, the first two and last two lines of context shall be ignored and another scan shall be made. Implementations may search more extensively for installation locations.

If no location can be found, the *patch* utility shall append the hunk to the reject file. The rejected hunk shall be written in context-difference format regardless of the format of the patch file. If the input was a normal or *ed*-style difference, the reject file may contain differences with zero lines of context. The line numbers on the hunks in the reject file may be different from the line numbers in the patch file since they shall reflect the approximate locations for the failed hunks in the new file rather than the old one.

If the type of patch is an *ed* diff, the implementation may accomplish the patching by invoking the *ed* utility.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

1 One or more lines were written to a reject file.

>1 An error occurred.

CONSEQUENCES OF ERRORS

Patches that cannot be correctly placed in the file shall be written to a reject file.

The following sections are informative.

APPLICATION USAGE

The **-R** option does not work with *ed* scripts because there is too little information to reconstruct the reverse operation.

The **-p** option makes it possible to customize a patch file to local user directory structures without manually editing the patch file. For example, if the filename in the patch file was:

/curds/whey/src/blurfl/blurfl.c

Setting **-p 0** gives the entire pathname unmodified; **-p 1** gives:

curds/whey/src/blurfl/blurfl.c

without the leading slash, **-p 4** gives:

blurfl/blurfl.c

and not specifying **-p** at all gives:

blurfl.c .

EXAMPLES

None.

RATIONALE

Some of the functionality in historical *patch* implementations was not specified. The following documents those features present in historical implementations that have not been specified.

A deleted piece of functionality was the **'+'** pseudo-option allowing an additional set of options and a patch file operand to be given. This was seen as being insufficiently useful to standardize.

In historical implementations, if the string **"Prereq:"** appeared in the header, the *patch* utility would search for the corresponding version information (the string specified in the header, delimited by <blank>s or the beginning or end of a line or the file) anywhere in the original file. This was deleted as too simplistic and insufficiently trustworthy a mechanism to standardize. For example, if:

Prereq: 1.2

were in the header, the presence of a delimited 1.2 anywhere in the file would satisfy the prerequisite.

The following options were dropped from historical implementations of *patch* as insufficiently useful to standardize:

-b The **-b** option historically provided a method for changing the name extension of the backup file from the default **.orig**. This option has been modified and retained in this volume of IEEE Std 1003.1-2001.

-F The **-F** option specified the number of lines of a context diff to ignore when searching for a place to install a patch.

-f The **-f** option historically caused *patch* not to request additional information from the user.

-r The **-r** option historically provided a method of overriding the extension of the reject file from the default **.rej**.

-s The **-s** option historically caused *patch* to work silently unless an error occurred.

-x The **-x** option historically set internal debugging flags.

In some file system implementations, the saving of a **.orig** file may produce unwanted results. In the case of 12, 13, or 14-character filenames (on file systems supporting 14-character maximum filenames), the **.orig** file overwrites the new file. The reject file may also exceed this filename limit. It was suggested, due to some historical practice, that a tilde (**'~'**) suffix be used instead of **.orig** and some other character instead of the **.rej** suffix. This was rejected because it is not obvious to the user which file is which. The suffixes **.orig** and **.rej** are clearer and more understandable.

The **-b** option has the opposite sense in some historical implementations-do not save the **.orig** file. The default case here is not to save the files, making *patch* behave more consistently with the other standard utilities.

The **-w** option in early proposals was changed to **-l** to match historical practice.

The **-N** option was included because without it, a non-interactive application cannot reject previously applied patches. For example, if a user is piping the output of *diff* into the *patch* utility, and the user only wants to patch a file to a newer version non-interactively, the **-N** option is required.

Changes to the **-l** option description were proposed to allow matching across <newline>s in addition to just <blank>s. Since this is not historical practice, and since some ambiguities could result, it is suggested that future developments in this area utilize another option letter, such as **-L**.

FUTURE DIRECTIONS

None.

SEE ALSO

ed , *diff*

COPYRIGHT