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** command reads a source (patch) file containing any of the three forms of difference (diff) listings produced by the **diff**(1) command (normal, context or in the style of **ed**(1)) and apply those differences to a file. By default, **patch** reads from the standard input.

patch attempts 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** will attempt to apply each of them as if they came from separate patch files. (In this case the name of the patch file must be determinable for each **diff** listing.)

OPTIONS

The following options are 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 will be overwritten; if multiple patches are applied to the same file, the **.orig** file will be written only for the first patch. When the –**o** *outfile* option is also specified, *file***.orig** will not be created but, if *outfile* already exists, *outfile***.orig** will be created.

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

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

–**D** *define* Mark changes with the C preprocessor construct:
#ifdef *define*
. . .
#endif

The option-argument *define* will be used as the differentiating symbol.

–**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 path name *patchfile*, rather than the standard input.

–**l** (The letter ell.) Cause any sequence of blank characters in the difference script to match any sequence of blank characters in the input file. Other characters will 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 are 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 will be applied to the intermediate versions of the file created by any previous patches, and will result in multiple, concatenated versions of the file being written to *outfile*.

−p *num* For all path names in the patch file that indicate the names of files to be patched, delete *num* path name components from the beginning of each path name. If the path name in the patch file is absolute, any leading slashes are considered the first component (that is, **−p 1** removes the leading slashes). Specifying **−p 0** causes the full path name to be used. If **−p** is not specified, only the basename (the final path name component) is 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. **patch** attempts to reverse each portion of the script before applying it. Rejected differences will 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 will be prompted to determine if the **−R** option should be set.

−r *rejectfile* Override the default reject filename. In the default case, the reject file will have the same name as the output file, with the suffix **.rej** appended to it. See **Patch Application**.

OPERANDS

The following operand is supported:

file A path name of a file to patch.

USAGE

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

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

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

Setting **−p 0** gives the entire path name 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.

When using −**b** 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 will overwrite the new file.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **patch**: **LC_CTYPE**, **LC_MESSAGES**, **LC_TIME**, and **NLSPATH**.

OUTPUT FILES

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

EXTENDED DESCRIPTION

A patchfile may contain patching instructions for more than one file; filenames are determined as specified in **Patch Determination**. When the −**b** option is specified, for each patched file, the original will 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 will be formed by appending the suffix **.rej** to the original filename.

Patchfile Format

The patch file must contain zero or more lines of header information followed by one or more patches. Each patch must 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.

patch recognizes 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 characters, **patch** will remove this sequence before proceeding. Within each patch, if the type of difference is context, **patch** recognizes the following expressions:

*** *filename timestamp* The patches arose from *filename*.

−−− *filename timestamp* The patches should be applied to *filename*.

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

Filename Determination

If no *file* operand is specified, **patch** performs the following steps to obtain a path name:

1. If the patch contains the strings *** and −−−, **patch** strips components from the beginning of each path name (depending on the presence or value of the −**p** option), then tests for the existence of both files in the current directory (or directory specified with the −**d** option).
2. If both files exist, **patch** assumes that no path name can be obtained from this step. If the header information contains a line with the string **Index:**, **patch** strips components from the beginning of the path name (depending on −**p**), then tests for the existence of this file in the current directory (or directory specified with the −**d** option).

3. If an **SCCS** directory exists in the current directory, **patch** will attempt to perform a **get** −**e SCCS/s.***filename* command to retrieve an editable version of the file.
4. If no path name can be obtained by applying the previous steps, or if the path names obtained do not exist, **patch** will write a prompt to standard output and request a filename interactively from standard input.

Patch Application

If the −**c**, −**e** or −**n** option is present, **patch** will 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, **patch** determines the type of difference based on the format of information within the hunk.

For each hunk, **patch** begins 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** scans 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 will take place, ignoring the first and last line of context. If that fails, the first two and last two lines of context will be ignored and another scan will be made. Implementations may search more extensively for installation locations.

If no location can be found, **patch** will append the hunk to the reject file. The rejected hunk will 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 will 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** command.

EXIT STATUS

The following exit values are returned:

0 Successful completion.
1 One or more lines were written to a reject file.
>1 An error occurred.

SEE ALSO

ed(1), **diff**(1), **environ**(5)