

NAME

sed – stream editor

SYNOPSIS

sed [**-n**] [**-e** *script*] [**-f** *sfile*] [*file*] ...

DESCRIPTION

Sed copies the named *files* (standard input default) to the standard output, edited according to a script of commands. The **-f** option causes the script to be taken from file *sfile*; these options accumulate. If there is just one **-e** option and no **-f**'s, the flag **-e** may be omitted. The **-n** option suppresses the default output.

A script consists of editing commands, one per line, of the following form:

[address [, address]] function [arguments]

In normal operation *sed* cyclically copies a line of input into a *pattern space* (unless there is something left after a 'D' command), applies in sequence all commands whose *addresses* select that pattern space, and at the end of the script copies the pattern space to the standard output (except under **-n**) and deletes the pattern space.

An *address* is either a decimal number that counts input lines cumulatively across files, a '\$' that addresses the last line of input, or a context address, 'regular expression', in the style of *ed*(1) modified thus:

The escape sequence '\n' matches a newline embedded in the pattern space.

A command line with no addresses selects every pattern space.

A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Editing commands can be applied only to non-selected pattern spaces by use of the negation function '!' (below).

In the following list of functions the maximum number of permissible addresses for each function is indicated in parentheses.

An argument denoted *text* consists of one or more lines, all but the last of which end with '\ ' to hide the newline. Backslashes in *text* are treated like backslashes in the replacement string of an 's' command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line.

An argument denoted *rfile* or *wfile* must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

(1) a\
text

Append. Place *text* on the output before reading the next input line.

(2) b *label*

Branch to the ':' command bearing the *label*. If *label* is empty, branch to the end of the script.

(2) c\
text

Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.

(2) d Delete the pattern space. Start the next cycle.

- (2)D Delete the initial segment of the pattern space through the first newline. Start the next cycle.
- (2)g Replace the contents of the pattern space by the contents of the hold space.
- (2)G Append the contents of the hold space to the pattern space.
- (2)h Replace the contents of the hold space by the contents of the pattern space.
- (2)H Append the contents of the pattern space to the hold space.
- (1)i\
text Insert. Place *text* on the standard output.
- (2)l List the pattern space on the standard output in an unambiguous form. Non-printing characters are spelled in two digit ascii, and long lines are folded.
- (2)n Copy the pattern space to the standard output. Replace the pattern space with the next line of input.
- (2)N Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)
- (2)p Print. Copy the pattern space to the standard output.
- (2)P Copy the initial segment of the pattern space through the first newline to the standard output.
- (1)q Quit. Branch to the end of the script. Do not start a new cycle.
- (2)r *rfile*
Read the contents of *rfile*. Place them on the output before reading the next input line.
- (2)s/*regular expression/replacement/flags*
Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character may be used instead of '/'. For a fuller description see *ed(1)*. *Flags* is zero or more of
 - g Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.
 - p Print the pattern space if a replacement was made.
 - w *wfile* Write. Append the pattern space to *wfile* if a replacement was made.
- (2)t *label*
Test. Branch to the ':' command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a 't'. If *label* is empty, branch to the end of the script.
- (2)w *wfile*
Write. Append the pattern space to *wfile*.
- (2)x Exchange the contents of the pattern and hold spaces.
- (2)y/*string1/string2/*
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.
- (2)! *function*
Don't. Apply the *function* (or group, if *function* is '{') only to lines *not* selected by the address(es).
- (0): *label*
This command does nothing; it bears a *label* for 'b' and 't' commands to branch to.
- (1)= Place the current line number on the standard output as a line.
- (2){ Execute the following commands through a matching '}' only when the pattern space is

selected.

- (0) An empty command is ignored.

SEE ALSO

ed(1), grep(1), awk(1)