SED(1) User Commands SED(1)

NAME

sed - stream editor for filtering and transforming text

SYNOPSIS

sed [OPTION]... {script-only-if-no-other-script} [input-file]...

DESCRIPTION

Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

-n, --quiet, --silent

suppress automatic printing of pattern space

-e script, --expression=script

add the script to the commands to be executed

-f script-file, --file=script-file

add the contents of script-file to the commands to be executed

-i[SUFFIX], --in-place[=SUFFIX]

edit files in place (makes backup if extension supplied)

-c, --copy

use copy instead of rename when shuffling files in -i mode

(avoids change of input file ownership)

-l N, --line-length=N

specify the desired line-wrap length for the 'l' command

--posix

disable all GNU extensions.

-r, --regexp-extended

use extended regular expressions in the script.

-s, --separate

consider files as separate rather than as a single continuous long stream.

-u, --unbuffered

load minimal amounts of data from the input files and flush the output buffers more often

--help display this help and exit

--version
output version information and exit

If no -e, --expression, -f, or --file option is given, then the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.

E-mail bug reports to: bonzini@gnu.org . Be sure to include the word ''sed'' somewhere in the ''Subject:'' field.

COMMAND SYNOPSIS

This is just a brief synopsis of sed commands to serve as a reminder to those who already know sed; other documentation (such as the texinfo

document) must be consulted for fuller descriptions.

Zero-address ''commands''

: label
Label for b and t commands.

#comment
The comment extends until the next newline (or the end of a -e script fragment).

} The closing bracket of a { } block.

Zero- or One- address commands

= Print the current line number.

a \

text Append text, which has each embedded newline preceded by a backslash.

i \

text Insert text, which has each embedded newline preceded by a backslash.

q Immediately quit the sed script without processing any more input, except that if auto-print is not disabled the current pattern space will be printed.

Q Immediately quit the sed script without processing any more input.

r filename
Append text read from filename.

R filename
Append a line read from filename.

Commands which accept address ranges

{ Begin a block of commands (end with a }).

b label

Branch to label; if label is omitted, branch to end of script.

t label
If a s/// has done a successful substitution since the last input line was read and since the last t or T command, then branch to label; if label is omitted, branch to end of script.

T label
If no s/// has done a successful substitution since the last input line was read and since the last t or T command, then branch to label; if label is omitted, branch to end of script.

c \

text Replace the selected lines with text, which has each embedded newline preceded by a backslash.

d Delete pattern space. Start next cycle.

D Delete up to the first embedded newline in the pattern space. Start next cycle, but skip reading from the input if there is still data in the pattern space.

h H Copy/append pattern space to hold space.

g G Copy/append hold space to pattern space.

x Exchange the contents of the hold and pattern spaces.

l List out the current line in a ''visually unambiguous'' form.

n N Read/append the next line of input into the pattern space.

p Print the current pattern space.

P Print up to the first embedded newline of the current pattern space.

s/regexp/replacement/
Attempt to match regexp against the pattern space. If successful, replace that portion matched with replacement. The replacement may contain the special character & to refer to that

portion of the pattern space which matched, and the special escapes \1 through \9 to refer to the corresponding matching sub-expressions in the regexp.

w filename
Write the current pattern space to filename.

W filename
Write the first line of the current pattern space to filename.

y/source/dest/
Transliterate the characters in the pattern space which appear in source to the corresponding character in dest.

Addresses

Sed commands can be given with no addresses, in which case the command will be executed for all input lines; with one address, in which case the command will only be executed for input lines which match that address; or with two addresses, in which case the command will be executed for all input lines which match the inclusive range of lines starting from the first address and continuing to the second address. Three things to note about address ranges: the syntax is addr1,addr2 (i.e., the addresses are separated by a comma); the line which addr1 matched will always be accepted, even if addr2 selects an earlier line; and if addr2 is a regexp, it will not be tested against the line that addr1 matched.

After the address (or address-range), and before the command, a ! may be inserted, which specifies that the command shall only be executed if the address (or address-range) does not match.

The following address types are supported:

number Match only the specified line number.

first~step
Match every step'th line starting with line first. For example, ''sed -n 1~2p'' will print all the odd-numbered lines in the input stream, and the address 2~5 will match every fifth line, starting with the second. (This is an extension.)

$ Match the last line.

/regexp/
Match lines matching the regular expression regexp.

\cregexpc
Match lines matching the regular expression regexp. The c may be any character.

GNU sed also supports some special 2-address forms:

0,addr2
Start out in "matched first address" state, until addr2 is found. This is similar to 1,addr2, except that if addr2 matches the very first line of input the 0,addr2 form will be at the end of its range, whereas the 1,addr2 form will still be at the beginning of its range.

addr1,+N
Will match addr1 and the N lines following addr1.

addr1,~N
Will match addr1 and the lines following addr1 until the next line whose input line number is a multiple of N.

REGULAR EXPRESSIONS

POSIX.2 BREs should be supported, but they aren't completely because of performance problems. The \n sequence in a regular expression matches the newline character, and similarly for \a, \t, and other sequences.

BUGS

E-mail bug reports to bonzini@gnu.org. Be sure to include the word ''sed'' somewhere in the ''Subject:'' field. Also, please include the output of ''sed --version'' in the body of your report if at all possible.

COPYRIGHT