NAME

sed – the stream editor

SYNOPSIS

sed [-n] [-g] [-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.

An -e option supplies a single edit command from the next argument; if there are several of these they are executed in the order in which they appear. If there is just one -e option and no -f 's, the -e flag may be omitted.

An -f option causes commands to be taken from the file "sfile"; if there are several of these they are executed in the order in which they appear; -e and -f commands may be mixed.

The -g option causes sed to act as though every substitute command in the script has a g suffix.

The -n option suppresses the default output.

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

```
[address [, address] ] function [arguments]
```

Normally sed cyclically copies a line of input into a current text buffer, then applies all commands whose addresses select the buffer in sequence, then copies the buffer to standard output and clears it.

The -n option suppresses normal output (so that only p and w output is done). Also, some commands (n, N) do their own line reads, and some others (d, D) cause all commands following in the script to be skipped (the D command also suppresses the clearing of the current text buffer that would normally occur before the next cycle).

It is also helpful to know that there's a second buffer (called the 'hold space' that can be copied or appended to or from or swapped with the current text buffer.

An address is: a decimal numeral (which matches the line it numbers where line numbers start at 1 and run cumulatively across files), or a '$' that addresses the last line of input, or a context address, which is a '/regular expression/', in the style of ed (1) modified thus:

(1) The escape sequence '\n' matches a newline embedded in the buffer, and '\t' matches a tab.

(2) A command line with no addresses selects every buffer.

(3) A command line with one address selects every buffer that matches that address.

(4) A command line with two addresses selects the inclusive range from the first input buffer that matches the first address through the next input buffer 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.) Once the second address is matched sed starts looking for the first one again; thus, any number of these ranges will be matched.

The negation operator '!' can prefix a command to apply it to every line not selected by the address(es).

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

An argument denoted "text" consists of one or more lines, with all but the last ending 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 whitespace (blanks and tabs) against the stripping that is done on every line of the script.

An argument denoted "rfile" or "wfile" must be last on the command line. Each wfile is created before processing begins. There can be at most 10 distinct wfile arguments.

a "text" (1)
  Append. Place text on output before reading the next input line.

b "label" (2)
: Branch to the ':' command bearing the label. If no label is given, branch to the end of the script.

c "text" (2)
: Change. Delete the current text buffer. With 0 or 1 address, or at the end of a 2-address range, place text on the output. Start the next cycle.

d (2)
: Delete the current text buffer. Start the next cycle.

D (2)
: Delete the first line of the current text buffer (all chars up to the first newline). Start the next cycle.

g (2)
: Replace the contents of the current text buffer with the contents of the hold space.

G (2)
: Append the contents of the hold space to the current text buffer.

h (2)
: Copy the current text buffer into the hold space.

H (2)
: Append a copy of the current text buffer to the hold space.

i "text" (1)
: Insert. Place text on the standard output.

l (2)
: List. Sends the pattern space to standard output. A "w" option may follow as in the s command below. Non-printable characters expand to:

```
  \b  --  backspace (ASCII 08)
  \t  --  tab       (ASCII 09)
  \n  --  newline   (ASCII 10)
  \r  --  return    (ASCII 13)
  \e  --  escape    (ASCII 27)
  \xx --  the ASCII character corresponding to 2 hex digits xx.
```

Dump. Hex-dump the pattern space to standard output.

n (2)
: Copy the current text buffer to standard output. Read the next line of input into it.

N (2)
: Append the next line of input to the current text buffer, inserting an embedded newline between the two. The current line number changes.

p (2)
: Print. Copy the current text buffer to the standard output.

P (2)
: Copy the first line of the current text buffer (all chars up to the first newline) to standard output.

q (1)
: Quit. Branch to the end of the script. Do not start a new cycle.

r "rfile" (1)
: Read the contents of rfile. Place them on the output before reading the next input line.

s /regular expression/replacement/flags (2)
: Substitute the replacement for instances of the regular expression in the current text buffer. Any character may be used instead of '/'. For a fuller description see ed (1). Flags is zero or more of the following:

  g -- Global. Substitute for all nonoverlapping instances of the string
  rather than just the first one.

p -- Print the pattern space if a replacement was made.

w -- Write. Append the current text buffer to a file argument as in a
w command if a replacement is made. Standard output is used if no
file argument is given

t "label" (2)
: Branch-if-test. Branch to the : command with the given label if any substitutes have been made since the most recent read of an input line or execution of a 't'or 'T'. If no label is given, branch to the end of the script.

T "label" (2)
: Branch-on-error. Branch to the : command with the given label if no substitutes have succeeded since the last input line or t or T command. Branch to the end of the script if no label is given.

w "wfile" (2)
: Write. Append the current text buffer to wfile .

W "wfile" (2)
: Write first line. Append first line of the current text buffer to wfile.

x (2)
: Exchange the contents of the current text buffer and hold space.

y /string1/string2/ (2)
: Translate. Replace each occurrence of a character in string1 with the corresponding character in string2. The lengths of these strings must be equal.

! "command" (2)
: All-but. Apply the function (or group, if function is '{') only to lines not selected by the address(es).

: "label" (0)
: This command does nothing but hold 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 current line matches the address or address range given.

An empty command is ignored.

PORTABILITY

This tool was reverse-engineered from BSD 4.1 UNIX sed, and (as far as the author's knowledge and tests can determine) is compatible with it. All documented features of BSD 4.1 sed are supported.

One undocumented feature (a leading 'n' in the first comment having the same effect as an -n command-line option) has been omitted.

The following bugs and limitations have been fixed:

* There is no hidden length limit (40 in BSD sed) on w file names.
* There is no limit (8 in BSD sed) on the length of labels.
* The exchange command now works for long pattern and hold spaces.

The following enhancements to existing commands have been made:

* a, i commands don't insist on a leading backslash-\n in the text.
* r, w commands don't insist on whitespace before the filename.
* The g, p and P options on s commands may be given in any order.

Some enhancements to regular-expression syntax have been made:

* \t is recognized in REs (and elswhere) as an escape for tab.

* In an RE, + calls for 1..n repeats of the previous pattern.

The following are completely new features:

* The l command (list, undocumented and weaker in BSD)

The 'L' command (hex dump).

* The W command (write first line of pattern space to file).

* The T command (branch on last substitute failed).

* Trailing comments are now allowed on command lines.

In addition, sed's error messages have been made more specific and informative.

The implementation is also significantly smaller and faster than BSD 4.1 sed. It uses only the standard I/O library and exit(3).

SEE ALSO

ed(1), grep(1), awk(1), lex(1), regexp(5)

AUTHOR

Eric S. Raymond <esr@snark.thyrsus.com> and Rene Rebe <rene@exactcode.de>. This program is distributed under the GPL.