NAME

sed – stream editor

SYNOPSIS

sed [–**n**] [–**e** *script*] [–**f** *sfilename*] [*filename*] . . .

DESCRIPTION

sed copies the *filename*s (standard input default) to the standard output, edited according to a script of commands.

OPTIONS

–**n** Suppress the default output.

–**e** *script* *script* is an edit command for **sed**. If there is just one –**e** option and no –**f** options, the –**e** flag may be omitted.

–**f** *sfilename* Take the script from *sfilename*.

USAGE

sed Scripts

sed scripts consist 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), sequentially applies all commands with *addresses* matching that pattern space until reaching the end of the script, copies the pattern space to the standard output (except under –**n**), and finally, deletes the pattern space.

Some commands use a *hold space* to save all or part of the pattern space for subsequent retrieval.

An *address* is either:

a decimal number linecount, which is cumulative across input files;

a **$**, which addresses the last input line;

or a context address, which is a **/***regular expression***/** as described on the **regexp**(5) manual page, with the following exceptions:

\?RE*?* In a context address, the construction \ *?regular expression?*, where *?* is any character, is identical to **/***regular expression***/**. Note: in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.

\n Matches a NEWLINE embedded in the pattern space.

. Matches any character except the NEWLINE ending the pattern space.

null A command line with no address selects every pattern space.

address Selects each pattern space that matches.

address1 , *address2*
Selects the inclusive range from the first pattern space matching *addrress1* to the first pattern space matching *address2*. Selects only one line if *address1* is greater than or equal to *address2*.

Comments

If the first nonwhite character in a line is a '**#**' (pound sign), **sed** treats that line as a comment, and ignores it. If, however, the first such line is of the form:

#n

sed runs as if the −**n** flag were specified.

Functions

The maximum number of permissible addresses for each function is indicated in parentheses in the list below.

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 SPACE and TAB characters against the stripping that is done on every script line.

An argument denoted *rfilename* or *wfilename* must terminate the command line and must be preceded by exactly one SPACE. Each *wfilename* is created before processing begins. There can be at most 10 distinct *wfilename* 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*. Branch to the end of the script if *label* is empty.

(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** *rfilename*
Read the contents of *rfilename*. 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 **regexp**(5). *flags* is zero or more of:

n — *n*= 1 – 512. Substitute for just the *n*th occurrence of the *regular* expression.

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 *wfilename* — Write: append the pattern space to *wfilename* 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** *wfilename*
Write: append the pattern space to *wfilename*.

(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* Do not: 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. Note: the maximum length of *label* is seven characters.

(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. Commands are separated by '**;**'.

(0) An empty command is ignored.

DIAGNOSTICS

Too many commands
The command list contained more than 200 commands.

Too much command text
The command list was too big for **sed** to handle. Text in the **a**, **c**, and **i** commands, text read in by **r** commands, addresses, regular expressions and

replacement strings in **s** commands, and translation tables in **y** commands all require **sed** to store data internally.

Command line too long
A command line was longer than 4000 characters.

Too many line numbers
More than 256 decimal number linecounts were specified as addresses in the command list.

Too many files in w commands
More than 10 different files were specified in **w** commands or **w** options for **s** commands in the command list.

Too many labels
More than 50 labels were specified in the command list.

Unrecognized command
A command was not one of the ones recognized by **sed**.

Extra text at end of command
A command had extra text after the end.

Illegal line number
An address was neither a decimal number linecount, a **$**, nor a context address.

Space missing before filename
There was no space between a **r** or **w** command, or the **w** option for a **s** command, and the filename specified for that command.

Too many {'s
There were more **{** than **}** in the list of commands to be executed.

Too many }'s
There were more **}** than **{** in the list of commands to be executed.

No addresses allowed
A command that takes no addresses had an address specified.

Only one address allowed
A command that takes one address had two addresses specified.

"\digit" out of range
The number in a *n* item in a regular expression or a replacement string in a **s** command was greater than 9.

Bad number
One of the endpoints in a range item in a regular expression (that is, an item of the form {*n*} or **{***n*,*m***}**) was not a number.

Range endpoint too large
One of the endpoints in a range item in a regular expression was greater than 255.

More than 2 numbers given in \{ \}
More than two endpoints were given in a range expression.

**} expected after **
A \\ appeared in a range expression and was not followed by a }.

First number exceeds second in \\{ \\}
The first endpoint in a range expression was greater than the second.

Illegal or missing delimiter
The delimiter at the end of a regular expression was absent.

\\(\\) imbalance
There were more **\\(** than **\\)**, or more **\\)** than **\\(**, in a regular expression.

[ ] imbalance
There were more **[** than **]**, or more **]** than **[**, in a regular expression.

First RE may not be null
The first regular expression in an address or in a **s** command was null (empty).

Ending delimiter missing on substitution
The ending delimiter in a **s** command was absent.

Ending delimiter missing on string
The ending delimiter in a **y** command was absent.

Transform strings not the same size
The two strings in a **y** command were not the same size.

Suffix too large - 512 max
The suffix in a **s** command, specifying which occurrence of the regular expression should be replaced, was greater than 512.

Label too long
A label in a command was longer than 8 characters.

Duplicate labels
The same label was specified by more than one **:** command.

File name too long
The filename specified in a **r** or **w** command, or in the **w** option for a **s** command, was longer than 1024 characters.

Output line too long.
An output line was longer than 4000 characters long.

Too many appends or reads after line *n*
More than 20 **a** or **r** commands were to be executed for line *n*.

Hold space overflowed.
More than 4000 characters were to be stored in the *hold space*.

FILES

usr/ucb/sed BSD sed

SEE ALSO

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

BUGS

There is a combined limit of 200 –**e** and –**f** arguments. In addition, there are various internal size limits which, in rare cases, may overflow. To overcome these limitations, either combine or break out scripts, or use a pipeline of **sed** commands.