NAME

sed – stream editor

SYNOPSIS

/usr/bin/sed [**–n**] *script* [*file . . .*]
/usr/bin/sed [**–n**] [**–e** *script*] . . . [**–f** *script_file*] . . . [*file . . .*]

/usr/xpg4/bin/sed [**–n**] *script* [*file . . .*]
/usr/xpg4/bin/sed [**–n**] [**–e** *script*] . . . [**–f** *script_file*] . . . [*file . . .*]

AVAILABILITY

/usr/bin/sed SUNWcsu

/usr/xpg4/bin/sed SUNWxcu4

DESCRIPTION

The **sed** command is a stream editor that reads one or more text files, makes editing changes according to a script of editing commands, and writes the results to standard output. The script is obtained from either the *script* operand string, or a combination of the option-arguments from the **–e** *script* and **–f** *script_file* options.

OPTIONS

The following options are supported;

–e *script* — *script* is an edit command for **sed**. See **USAGE** below for more information on the format of *script*. If there is just one **–e** option and no **–f** options, the flag **–e** may be omitted.

–f *script_file* — Take the script from *script_file*. *script_file* consists of editing commands, one per line.

–n — Suppress the default output.

Multiple **–e** and **–f** options may be specified. All commands are added to the script in the order specified, regardless of their origin.

OPERANDS

The following operands are supported:

file — A path name of a file whose contents will be read and edited. If multiple *file* operands are specified, the named files will be read in the order specified and the concatenation will be edited. If no *file* operands are specified, the standard input will be used.

script — A string to be used as the script of editing commands. The application must not present a *script* that violates the restrictions of a text file except that the final character need not be a NEWLINE character.

USAGE

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

[*address* **[,** *address* **]]** *function* **[** *arguments* **]**

Zero or more blank characters are accepted before the first address and before *command.* Any number of semicolons are accepted before the first address.

In normal operation, **sed** cyclically copies a line of input (less its terminating NEWLINE character) 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 copies the resulting pattern space to the standard output (except under –**n**) and deletes the pattern space. Whenever the pattern space is written to standard output or a named file, **sed** will immediately follow it with a NEWLINE character.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval. The *pattern* and *hold spaces* will each be able to hold at least **8192** bytes.

sed Addresses

An *address* is either empty, a decimal number that counts input lines cumulatively across files, a **$** that addresses the last line of input, or a context address, which consists of a **/***regular expression***/** as described on the **regexp**(5) manual page.

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 address. Thereafter the process is repeated, looking again for the first address. (If the second address is a number less than or equal to the line number selected by the first address, only the line corresponding to the first address is selected.)

Typically, address are separated from each other by a comma (,). They may also be separated by a semicolon (;).

sed Regular Expressions

sed supports the basic regular expressions described on the **regexp**(5) manual page, with the following additions:

cREc In a context address, the construction *cREc*, where *c* is any character other than a backslash or NEWLINE character, is identical to /*RE*/. If the character designated by *c* appears following a backslash, then it is considered to be that literal character, which does not terminate the RE. For example, in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.

\n The escape sequence **\n** matches a NEWLINE character embedded in the pattern space. A literal NEWLINE character must not be used in the regular expression of a context address or in the substitute command.

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

sed Editing Commands

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

The **r** and **w** commands take an optional *rfile* (or *wfile*) parameter, separated from the command letter by one or more blank characters; implementations may allow zero separation as an extension.

Multiple commands can be specified by separating them with a semicolon (;) on the same command line.

The *text* argument consists of one or more lines, all but the last of which end with **** to hide the NEWLINE. Each embedded NEWLINE character in the text must be preceded by a backslash. Other backslashes in text are removed and the following character is treated literally. 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. The *rfile* or *wfile* argument must terminate the command line and must be preceded by exactly one blank. The use of the *wfile* parameter causes that file to be initially created, if it does not exist, or will replace the contents of an existing file. There can be at most 10 distinct *wfile* arguments.

Regular expressions match entire strings, not just individual lines, but a NEWLINE character is matched by **\n** in a **sed** RE; a NEWLINE character is not allowed in an RE. Also note that **\n** cannot be used to match a NEWLINE character at the end of an input line; NEWLINE characters appear in the pattern space as a result of the **N** editing command.

Two of the commands take a *command-list*, which is a list of **sed** commands separated by NEWLINE characters, as follows:

{ *command*
command
}

The **{** can be preceded with blank characters and can be followed with white space. The *commands* can be preceded by white space. The terminating **}** must be preceded by a NEWLINE character and can be preceeded or followed by <blank>s. The braces may be preceeded or followed by <blank>s. The command may be preceeded by <blank>s, but may not be followed by <blank>s.

The following table lists the functions.

Maximum Number of Addresses	Command	Description
2	{*command-list* }	Execute *command-list* only when the pattern space is selected.
1	**a** *text*	Append by executing **N** command or beginning a new cycle. 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. Labels are recognized unique up to eight characters.
2	**c** *text*	Change. Delete the pattern space. 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 new-line. Start the next cycle. (See the **N** command below.)
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-printable characters are displayed in octal notation and long lines are folded. The characters (\\\\, \\a, \\b, \\f, \\r, \\t, and \\v) are written as the corresponding escape sequences. Non-printable characters not in that table will be written as one three-digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first). If the size of a byte on the system is greater than nine bits, the format used for non-printable characters is implementation-dependent. Long lines will be folded, with the point of folding indicated by writing a backslash followed by a newline character; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line will be marked with a **$**.

Maximum Number of Addresses	Command	Description
2	**n**	Copy the pattern space to the standard output if default output is not supressed. 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 new-line. (The current line number changes.) If no next line of input is available, the **N** command verb shall branch to the end of the script and quit without starting a new cycle and without writing the pattern space.
2	**p**	Print. Copy the pattern space to the standard output.
2	**P**	Copy the initial segment of the pattern space through the first new-line 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. If *rfile* does not exist or cannot be read, it is treated as if it were an empty file, causing no error condition.
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*. The first occurrence of **w** will cause *wfile* to be cleared. Subsequent invocations of **w** will append. Each time the **sed** command is used, *wfile* is overwritten.
2	**x**	Exchange the contents of the pattern and hold spaces.
2	**!** *command*	Don't. Apply the *command* (or group, if *command* 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.
0	**#**	If a **#** appears as the first character on a line of a script file, then that entire line is treated as a comment, with one exception: if a **#** appears on the first line and the character after the **#** is an **n**, then the default output will be suppressed. The rest of the line after **#n** is also ignored. A script file must contain at least one non-comment line.

Maximum Number of Addresses	**Command** (Using *strings*) and **Description**
2	**s**/*regular expression*/*replacement*/*flags* Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character other than backslash or newline can be used instead of a slash to delimit the RE and the replacement. Within the RE and the replacement, the RE delimiter itself can be used as a literal character if it is preceded by a backslash. An ampersand (**&**) appearing in the *replacement* will be replaced by the string matching the RE. The special meaning of **&** in this context can be suppressed by preceding it by backslash. The characters *n*, where *n* is a digit, will be replaced by the text matched by the corresponding backreference expression. For each backslash (\\) encountered in scanning *replacement* from beginning to end, the following character loses its special meaning (if any). It is unspecified what special meaning is given to any character other than &, \\ or digits. A line can be split by substituting a newline character into it. The application must escape the newline character in the *replacement* by preceding it by backslash. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces. *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. If both *g* and *n* are specified, the results are unspecified. **p** Print the pattern space if a replacement was made. **P** Copy the initial segment of the pattern space through the first new-line to the standard output. **w** *wfile* Write. Append the pattern space to *wfile* if a replacement was made. The first occurrence of **w** will cause *wfile* to be cleared. Subsequent invocations of **w** will append. Each time the **sed** command is used, *wfile* is overwritten.
2	**y**/ *string1* / *string2* / Transform. Replace all occurrences of characters in *string1* with the corresponding characters in *string2*. *string1* and *string2* must have the same number of characters, or if any of the characters in *string1* appear more than once, the results are undefined. Any character other than backslash or newline can be used instead of slash to delimit the strings. Within *string1* and *string2*, the delimiter itself can be used as a literal character if it is preceded by a backslash. For example, **y**/abc/ABC/ replaces a with A, b with B, and c with C.

EXAMPLES

This **sed** script simulates the BSD **cat** –**s** command, squeezing excess blank lines from standard input.

```
sed -n '
# Write non-empty lines.
/./    {
       p
       d
       }
# Write a single empty line, then look for more empty lines.
/^$/       p
# Get next line, discard the held <newline> (empty line),
# and look for more empty lines.
:Empty
/^$/       {
       N
       s/.//
       b Empty
       }
# Write the non-empty line before going back to search
# for the first in a set of empty lines.
       p
'
```

ENVIRONMENT

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

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

awk(1), **ed**(1), **grep**(1), **environ**(5), **regexp**(5)