A

Quick Reference for sed

Command-Line Syntax

The syntax for invoking sed has two forms:

sed [*-n*][*-e*] '*command*' *file(s)*
sed [*-n*] *-f scriptfile file(s)*

The first form allows you to specify an editing command on the command line, surrounded by single quotes. The second form allows you to specify a *scriptfile*, a file containing sed commands. Both forms may be used together, and they may be used multiple times. The resulting editing script is the concatenation of the commands and script files.

The following options are recognized:

-n
: Only print lines specified with the **p** command or the **p** flag of the **s** command.

-e cmd
: Next argument is an editing command. Useful if multiple scripts are specified.

-f *file*
: Next argument is a file containing editing commands.

If the first line of the script is "#n", sed behaves as if *-n* had been specified.

Frequently used sed scripts are usually invoked from a shell script. Since this is the same for sed or awk, see the section "Shell Wrapper for Invoking awk" in Appendix B, *Quick Reference for awk*.

Syntax of sed Commands

Sed commands have the general form:

[*address*[,*address*]][!]*command* [*arguments*]

Sed copies each line of input into a pattern space. Sed instructions consist of addresses and editing commands. If the address of the command matches the line in the pattern space, then the command is applied to that line. If a command has no address, then it is applied to each input line. If a command changes the contents of the space, subsequent command-addresses will be applied to the current line in the pattern space, not the original input line.

Pattern Addressing

address can be either a line number or a *pattern*, enclosed in slashes (/*pattern*/). A pattern is described using a regular expression. Additionally, \n can be used to match any newline in the pattern space (resulting from the **N** command), but not the newline at the end of the pattern space.

If no pattern is specified, the command will be applied to all lines. If only one address is specified, the command will be applied to all lines matching that address. If two comma-separated addresses are specified, the command will be applied to a range of lines between the first and second addresses, inclusively. Some commands accept only one address: **a**, **i**, **r**, **q**, and **=**.

The ! operator following an address causes sed to apply the command to all lines that do not match the address.

Braces ({}) are used in sed to nest one address inside another or to apply multiple commands at the same address.

```
[/pattern/[,/pattern/]]{
command1
command2
}
```

The opening curly brace must end a line, and the closing curly brace must be on a line by itself. Be sure there are no spaces after the braces.

Regular Expression Metacharacters for sed

The following table lists the pattern-matching metacharacters that were discussed in Chapter 3, *Understanding Regular Expression Syntax*.

Note that an empty regular expression "//" is the same as the previous regular expression.

Table A–1: Pattern-Matching Metacharacters

Special Characters	Usage
.	Matches any single character except *newline*.
*	Matches any number (including zero) of the single character (including a character specified by a regular expression) that immediately precedes it.
[...]	Matches any one of the class of characters enclosed between the brackets. All other metacharacters lose their meaning when specified as members of a class. A circumflex (^) as the first character inside brackets reverses the match to all characters except newline and those listed in the class. A hyphen (-) is used to indicate a range of characters. The close bracket (]) as the first character in the class is a member of the class.
\{*n,m*\}	Matches a range of occurrences of the single character (including a character specified by a regular expression) that immediately precedes it. \{*n*\} will match exactly *n* occurrences, \{*n,*\} will match at least *n* occurrences, and \{*n,m*\} will match any number of occurrences between *n* and *m*. (sed and grep only).
^	Locates regular expression that follows at the beginning of line. The ^ is only special when it occurs at the beginning of the regular expression.
$	Locates preceding regular expression at the end of line. The $ is only special when it occurs at the end of the regular expression.
\	Escapes the special character that follows.
\(\)	Saves the pattern enclosed between "\(" and "\)" into a special holding space. Up to nine patterns can be saved in this way on a single line. They can be "replayed" in substitutions by the escape sequences "\1" to "\9".
n	Matches the *n*th pattern previously saved by "\(" and "\)", where *n* is a number from 1 to 9 and previously saved patterns are counted from the left on the line.
&	Prints the entire matched text when used in a replacement string.

Command Summary for sed

: :*label*

Label a line in the script for the transfer of control by **b** or **t**. *label* may contain up to seven characters. (The POSIX standard says that an implementation can allow longer labels if it wishes to. GNU sed allows labels to be of any length.)

= [*address*]**=**
Write to standard output the line number of addressed line.

a [*address*]**a**
text

Append *text* following each line matched by *address*. If *text* goes over more than one line, newlines must be "hidden" by preceding them with a backslash. The *text* will be terminated by the first newline that is not hidden in this way. The *text* is not available in the pattern space and subsequent commands cannot be applied to it. The results of this command are sent to standard output when the list of editing commands is finished, regardless of what happens to the current line in the pattern space.

b [*address1*[,*address2*]]**b**[*label*]
Transfer control unconditionally (branch) to :*label* elsewhere in script. That is, the command following the *label* is the next command applied to the current line. If no *label* is specified, control falls through to the end of the script, so no more commands are applied to the current line.

c [*address1*[,*address2*]]**c**
text

Replace (change) the lines selected by the address with *text*. When a range of lines is specified, all lines as a group are replaced by a single copy of *text*. The newline following each line of *text* must be escaped by a backslash, except the last line. The contents of the pattern space are, in effect, deleted and no subsequent editing commands can be applied to it (or to *text*).

d [*address1*[,*address2*]]**d**
Delete line(s) from pattern space. Thus, the line is not passed to standard output. A new line of input is read and editing resumes with first command in script.

D [*address1*[,*address2*]]**D**
Delete first part (up to embedded newline) of multiline pattern space created by **N** command and resume editing with first command in script. If this command empties the pattern space, then a new line of input is read, as if the **d** command had been executed.

g [*address1*[,*address2*]]**g**
Copy (get) contents of hold space (see **h** or **H** command) into the pattern space, wiping out previous contents.

G [*address1*[,*address2*]]**G**
Append newline followed by contents of hold space (see **h** or **H** command) to contents of the pattern space. If hold space is empty, a newline is still appended to the pattern space.

h [*address1*[,*address2*]]**h**
Copy pattern space into hold space, a special temporary buffer. Previous contents of hold space are wiped out.

H [*address1*[,*address2*]]**H**
Append newline and contents of pattern space to contents of the hold space. Even if hold space is empty, this command still appends the newline first.

i [*address1*]**i**
text

Insert *text* before each line matched by *address*. (See **a** for details on *text*.)

l [*address1*[,*address2*]]**l**
List the contents of the pattern space, showing nonprinting characters as ASCII codes. Long lines are wrapped.

n [*address1*[,*address2*]]**n**
Read next line of input into pattern space. Current line is sent to standard output. New line becomes current line and increments line counter. Control passes to command following **n** instead of resuming at the top of the script.

N [*address1*[,*address2*]]**N**
Append next input line to contents of pattern space; the new line is separated from the previous contents of the pattern space by a newline. (This command is designed to allow pattern matches across two lines. Using \n to match the embedded newline, you can match patterns across multiple lines.)

p [*address1*[,*address2*]]**p**
Print the addressed line(s). Note that this can result in duplicate output unless default output is suppressed by using "#n" or the *-n* command-line option. Typically used before commands that change flow control (**d**, **n**, **b**) and might prevent the current line from being output.

P [*address1*[,*address2*]]**P**
Print first part (up to embedded newline) of multiline pattern space created by **N** command. Same as **p** if **N** has not been applied to a line.

q [*address*]**q**
Quit when *address* is encountered. The addressed line is first written to output (if default output is not suppressed), along with any text appended to it by previous **a** or **r** commands.

r [*address*]**r** *file*
Read contents of *file* and append after the contents of the pattern space. Exactly one space must be put between **r** and the filename.

s [*address1*[,*address2*]]**s**/*pattern*/*replacement*/[*flags*]
Substitute *replacement* for *pattern* on each addressed line. If pattern addresses are used, the pattern **//** represents the last pattern address specified. The following flags can be specified:

- **n** Replace *n*th instance of /*pattern*/ on each addressed line. *n* is any number in the range 1 to 512, and the default is 1.
- **g** Replace all instances of /*pattern*/ on each addressed line, not just the first instance.
- **p** Print the line if a successful substitution is done. If several successful substitutions are done, multiple copies of the line will be printed.
- **w** *file* Write the line to *file* if a replacement was done. A maximum of 10 different *files* can be opened.

t [*address1*[,*address2*]]**t** [*label*]
Test if successful substitutions have been made on addressed lines, and if so, branch to line marked by :*label*. (See **b** and :.) If label is not specified, control falls through to bottom of script.

w [*address1*[,*address2*]]**w** *file*
Append contents of pattern space to *file*. This action occurs when the command is encountered rather than when the pattern space is output. Exactly one space must separate the **w** and the filename. A maximum of 10 different files can be opened in a script. This command will create the file if it does not exist; if the file exists, its contents will be overwritten each time the script is executed. Multiple write commands that direct output to the same file append to the end of the file.

x [*address1*[,*address2*]]**x**
Exchange contents of the pattern space with the contents of the hold space.

y [*address1*[,*address2*]]**y**/*abc*/*xyz*/
Transform each character by position in string *abc* to its equivalent in string *xyz*.