Sed, The Unix Stream Editor

“*Should be part of every gentlemans toolbox*” ME

Contents

The gnu sed has some features not found in other versions of sed. But many of the examples should work for all versions of sed.
Sed is actually a virtual machine, which has 2 registers, the 'work-space' and the 'hold-space' and a set of “instructions” or commands which manipulate these two registers.
This document contains an almost complete copy of the “sed oneliners” document available at sed.sf.net/sed1line.htm since that document is so complete and well compiled that there is not much point reinventing the wheel. I have only reformatted the document slightly so that it can be compiled into pdf using the scripts at bumble.sf.net/books/fo book

`http://www.grymoire.com/Unix/Sed.html`
: A concise introduction

`http://sed.sourceforge.net/`
: The “home page” for sed.

`http://sed.sourceforge.net/sedfaq.html`
: comprehensive recipes

`http://sed.sourceforge.net/sed1line.txt`
: a long list of one line sed scripts covering all the advanced techniques

Transliterate characters

```
sed "y/abcd/ABCD/g"
```

Section 1

Books About Sed

`sed`
: and awk, 2nd Edition Dale Dougherty and Arnold Robbins (O'Reilly, 1997; http://www.ora.com),

`UNIX`
: Text Processing Dale Dougherty and Tim O'Reilly (Hayden Books, 1987)

`the`
: tutorials by Mike Arst distributed in U-SEDIT2.ZIP.

`"Mastering`
: Regular Expressions” Jeffrey Friedl (O'Reilly, 1997).

Section 2

History

The sed stream editor has been around since the very early days of unix. One of its original motivations was that the computer in use (pdp?) did not have the capability to load large text files into memory in order to edit them "interactively".

Section 3

Getting Help For Sed

View the man page for sed

```
man sed
```

View the man page for the line editing program 'ed' a forerunner of 'sed'

```
man ed
```

View information about regular expression

```
man regexp
```

Section 4

Simple Usage

Sed takes one or more editing commands and applies all of them, in sequence, to each line of input. After all the commands have been applied to the first input line, that line is output and a second input line is taken for processing, and the cycle repeats. The preceding examples assume that input comes from the standard input device (i.e, the console, normally this will be piped input). One or more filenames can be appended to the command line if the input does not come from stdin. Output is sent to stdout (the screen). Thus

```
cat filename | sed '10q'          ~( uses piped input )
```

```
sed '10q' filename                ~( same effect, avoids a useless "cat"
  ⇒ )
```

```
sed '10q' filename > newfile      ~( redirects output to disk )
```

Section 5

Gotchas

These are tricky things that can cause the poor uncautious sed-user head-scratching beffudlement and a pervading existential 'ennui'.

Check for 'dos' carriage return characters '\r' which are not desireable

```
sed -n l file.txt      ~('l' prints unprintable characters visibly)
```

```
sed -n 'l' file.txt    ~(the same)
```

If you saw any '\r' characters in the previous, then get rid of them

```
fromdos file.txt     ~(those '\r' can cause annoying silent bugs)
```

```
dos2unix file.txt    ~(another way to say the same thing, I think)
```

Duplicate words, using groupings and backreferences.

```
s/\([a-z]+\)/\1\1/g;
```

```
s/([a-z]+)/\1\1/g;    ~(gnu sed with the -r flag, this is 'gotcha')
```

The 'e' gnu command seems to have to be the last thing on the line

```
echo ls | "e;p"      ~(doesnt work)
```

```
echo ls | "e"        ~(works)
```

The 'q' quit command can only have one address

```
1,4 q       ~(doesnt work)
```

If a one-line script contains a "!", it must use single quotes not double

```
sed -n "/^ *#/!p"   ~(Incorrect: the bash shell interprets the "!"
  ⇒ character)
```

If an 'a' (append) or 'i' (insert) command has a space after the backslash (in any line) then the command doesnt work. This is a really annoying one. use 'sed -i "s/\s\+$//" scriptname' to fix this problem

'd' delete immediately starts a new cycle

```
One might try not to print anything after a line containing
"-end-" including the line itself, with the following command
 /-end-/ { d;q }
BUT it doesnt work because the 'q' command will never be executed
owing to the behavior of the 'd' delete command
```

When putting a ']' character in brackets, dont escape it

```
s/\[[^]]*\]//g NOT s/\[[^\]]*\]//g   ~(at least in gnu sed )
```

Section 6

Scripts

Run a sed script in a file

```
sed -f script file
```

```
sed --file=script file       ~(the same)
```

```
cat file | sed -f script     ~(the same)
```

Find out where your sed executable is

```
which sed
```

Create an interpreting script with extended regexes (-r)

```
#!/bin/sed -rf    ~(this is the first script line, try 'which sed')
```

```
./script          ~(the script can be run with just its name)
```

Section 7

Sed Commands

sed command summary

d	Delete the pattern space, start the next cycle
p	Print the pattern space
q 4	Quit the script with exit code '4'
e 'ls'	Execute the shell 'ls' command and put in the pattern space
e	Execute the shell command in the pattern space
a\	Append text, each line ending in '\' except the last
i\	Insert text, each line ending in '\' except the last
D	Delete text in the pattern space up to the first newline.
N	Add a newline and the next line of input to the pattern space.
P	Print out the portion of the pattern space up to the first newline.
h	Replace the hold space with the pattern space.
H	Append a newline and the pattern space to the hold space.
g	Replace the pattern space with the hold space.
G	Append a newline and the hold space to pattern space.
x	Exchange the contents of the hold and pattern spaces.

modifiers

\L	Turn the replacement to lowercase until a \U or \E is found,
\l	Turn the next character to lowercase,
\U	Turn the replacement to uppercase until a \L or \E is found,
\u	Turn the next character to uppercase,
\E	Stop case conversion started by \L or \U.

Add 2 lines of text at the end of the file

```
$ a\
added text\
last line
```

Section 8

Multiple Commands

Either use the -e switch or semi-colons ';' for multiple commands

```
sed -e 's/a/A/' -e 's/b/B/' <textfile
```

```
sed 's/a/A/; s/b/B/' <textfile    ~(the same)
```

Section 9

Making Changes To Files

Change 'frog' for 'toad' and save the changes to the file 'newfile'

```
sed 's/frog/toad/g' < oldfile > newfile
```

This does NOT work, the file 'textfile' get truncated

```
sed 's/frog/toad/g' < textfile > textfile ~(use the -i switch instead)
```

Change yes to no in all files ending in '.txt' and back up to .bak

```
sed -i.bak "s/yes/no/g" *.txt     ~(gnused 4)
```

Section 10

Command Line Switches

Print only the lines in a file which contain the word 'sky'

```
sed -n 's/sky/&/p' <file
```

Delete the word 'tree' in the files f1, f2, and f3

```
sed 's/tree//g' f1 f2 f3
```

Run the sed script 'script'

```
sed -f script <textfile
```

Create an interpreting script with extended regexes (-r)

```
#!/bin/sed -rf   ~(this is the first line of the script, try 'which sed
  ⇒ ')
```

```
./script          ~(the script can be run with just its name)
```

Create an interpreting script which doesnt print the input lines

```
~(!/bin/sed -nf    ##(this may not work in some versions of sed)
```

Section 11

Substition Switches

Delete only the second match of the word 'big' on each line

```
sed 's/big//2' <file
```

Delete all occurrances of the word 'big' in a file

```
sed 's/big//g' <file
```

Delete the the 2nd and following instances of the word 'cat' on each line

```
sed 's/cat//2g' <file    ~(doesnt delete the first 'cat' on each line)
```

Delete the 80th character on each line

```
sed 's/.//80' <file    ~(the number must be less than 512)
```

Section 12

Sed Commands

Print the first 100 lines of a file

```
sed '100q' test
```

12.1 Inserting Text

Insert 2 lines of text before the first line of the file

```
1i\
<html> \
<head> \
```

Section 13

Writing To A File

Write all lines which contain the word 'sky' to the file 'words'

```
sed -n 's/sky/&/w words' < file   ~(only one space between w and 'words
  ⇒ ')
```

Section 14

Using What Was Matched

Duplicate lowercase letters

```
sed 's/[a-z]/&/'
```

Use a backreference

```
sed 's/\([a-z]*\).*/\1/'
```

Using the matched text in a range

```
echo abcd  | sed '/abc/s//x/;'   ~(prints 'xd')
```

(this is a trick from the days of the unix 'ed' text editor)

Section 15

Case Insensitive Matches

Replace 'this' or 'ThIS' (etc) with 'that'

```
/this/I s/this/that/i   ~(gnu sed and some others)
```

Section 16

Gnu Sed

`http://www.gnu.org/software/sed/manual/html_node/`
gnused documentation

Detailed information about gnu regular expressions (used in sed)

```
man 7 regex
```

The -r switch in gnused removes the necessity for some backslashes '\'

```
sed -r 'b{3,}'     ~(matches 3 or more 'b's)
```

```
sed 'b\{3,\}'      ~(the same, without the '-r' switch)
```

Read in the result of a shell command (with the 'e' command

```
echo ls | sed "e"    ~(prints the contents of the folder)
```

Execute shell commands in 'com.txt' on lines beginning with '!'

```
sed "/^!/ { s/!//;e }" com.txt ~(not working)
```

Regular Expression Examples

examples without the '-r' switch

`b\{4\}`	Matches 4 'b's
`a\{3,\}`	Matches 3 or more 'a's
`q\+`	Matches one or more 'q's
`\(big|small\)`	Matches 'big' or 'small'

Extended Patterns

18.1 Matching Words

Convert words beginning in 'b' or 'c' to upper case

```
sed -r "s/\<(b|c)[a-z]+/\U&/g"
```

Convert upper case words lower case

```
sed -r "s/\<[A-Z]+\>/\L&/g"   ~(gnused, not very international)
```

Match word boundaries, words beginning in 'a' or 'b'

```
sed -r "s/\<(a|b)[a-z]+/&/g"
```

18.2 Alternation

Delete the words red, green, blue

```
sed 's/red\|green\|blue//g'          ~(gnu sed)
sed -r 's/red|green|blue//g'         ~(the -r switch changes the syntax)
sed 's/red//g;s/green//g;s/blue//g   ~(other versions of sed)
```

18.3 International Character Classes

Delete uppercase letters or the letter 'a', 'b'

```
sed -r 's/[[:upper:]ab]//g'
```

gnused character classes (for international text)

`[[:alnum:]]`	[A-Za-z0-9] Alphanumeric characters
`[[:alpha:]]`	[A-Za-z] Alphabetic characters
`[[:blank:]]`	[\x09] Space or tab characters only
`[[:cntrl:]]`	[\x00-\x19\x7F] Control characters
`[[:digit:]]`	[0-9] Numeric characters
`[[:graph:]]`	[!-~] Printable and visible characters
`[[:lower:]]`	[a-z] Lower-case alphabetic characters
`[[:print:]]`	[-~] Printable (non-Control) characters
`[[:punct:]]`	[!-/:-@[-`{-~] Punctuation characters
`[[:space:]]`	[\t\v\f] All whitespace chars
`[[:upper:]]`	[A-Z] Upper-case alphabetic characters
`[[:xdigit:]]`	[0-9a-fA-F] Hexadecimal digit characters

the special character classes in short form

`\s`	Space characters
`\w`	A word character
`and others ...`	

Section 19

Uppercase And Lowercase

The following may only apply to gnu sed.

Turn to lower case all lines beginning with a '+' plus sign

```
sed '/^ *+$/s/.*/\L&/g'
```

```
sed '/^\s*+$/s/.*/\L&/g'                ~(the same, with some 'seds' like
  ⇒ gnu)
```

```
sed '/^[[:space:]]*+$/s/.*/\L&/g'    ~(the same, again for gnused)  )
```

Make all words in the line lower case (so 'TrEE' becomes 'tree')

```
sed "s/\w\+/\L&/g"
```

```
sed -r "s/\w+/\L&/g"  ~(the same)
```

```
sed -r "s/\w/\l&/g"   ~(the same)
```

Make all words 'capital case' (that is 'bumBle' becomes 'Bumble')

```
sed -r "s/(\w)(\w*)/u\1L\2/g"
```

```
sed "s/\(\w\)\(\w*\)/u\1L\2/g"    ~(the same)
```

Section 20

Sed One Line Scripts

The following is a slight reformatting of the one line sed script document by Eric Pement - pemente[at]northpark[dot]e
(version 5.5 Dec. 29, 2005)
Latest version of the original document (in English) is usually at: http://sed.sourceforge.net/sed1line.txt http://www

Section 21

White Space

'squeeze' multiple spaces into one on every line in the file

```
sed "s/ \+/ /g" doc.txt      ~(only spaces, not tab characters ...)
```

```
sed -r "s/ +/ /g" doc.txt    ~(exactly the same)
```

Squeeze all space characters (including tabs) into one space character

```
sed "s/\s\+/ /g" doc.txt     ~(not all seds)  )
```

```
sed -r "s/\s+/ /g" doc.txt   ~(the same, gnu sed)
```

```
sed -r "s/[[:space:]]+/ /g" doc.txt   ~(the same, gnu sed)
```

```
sed -r "s/[[:blank:]]+/ /g" doc.txt   ~(almost the same, gnu sed)
```

Section 22

File Spacing

Double space a file

```
sed G
```

Double space a file which already has blank lines in it. Output file Should contain no more than one blank line between lines of text.

```
sed '/^$/d;G'
```

Triple space a file

```
sed 'G;G'
```

Undo double-spacing (assumes even-numbered lines are always blank)

```
sed 'n;d'
```

Insert a blank line above every line which matches "regex"

```
sed '/regex/{x;p;x;}'
```

Insert a blank line below every line which matches "regex"

```
sed '/regex/G'
```

Insert a blank line above and below every line which matches "regex"

```
sed '/regex/{x;p;x;G;}'
```

Section 23

Numbering

Number each line of a file (simple left alignment). Using a tab (see Note on '\t' at end of file) instead of space will preserve margins.

```
sed = filename | sed 'N;s/\n/\t/'
```

Number each line of a file (number on left, right-aligned)

```
sed = filename | sed 'N; s/^/     /; s/ *\(.\{6,\}\)\n/\1  /'
```

Number each line of file, but only print numbers if line is not blank

```
sed '/./=' filename | sed '/./N; s/\n/ /'
```

Count lines (emulates "wc -l")

```
sed -n '$='
```

Section 24

Text Conversion And Substitution

IN UNIX ENVIRONMENT: convert DOS newlines (CR/LF) to Unix format.

```
sed 's/.$//'                    ~( assumes that all lines end with CR/LF )
sed 's/^M$//'                   ~( in bash/tcsh, press Ctrl-V then Ctrl-M )
sed 's/\x0D$//'                 ~( works on ssed, gsed 3.02.80 or higher )
```

IN UNIX ENVIRONMENT: convert Unix newlines (LF) to DOS format.

```
sed "s/$/`echo -e \\\r`/"             ~( command line under ksh )
sed 's/$'"/`echo \\\r`/"              ~( command line under bash )
sed "s/$/`echo \\\r`/"                ~( command line under zsh )
sed 's/$/\r/'                         ~( gsed 3.02.80 or higher )
```

IN DOS ENVIRONMENT: convert Unix newlines (LF) to DOS format.

```
sed "s/$//"                           ~( method 1 )
sed -n p                              ~( method 2 )
```

IN DOS ENVIRONMENT: convert DOS newlines (CR/LF) to Unix format. Can only be done with UnxUtils sed, version 4.0.7 or higher. The UnxUtils version can be identified by the custom "–text" switch Which appears when you use the "–help" switch. Otherwise, changing DOS newlines to Unix newlines cannot be done with sed in a DOS Environment. Use "tr" instead.

```
sed "s/\r//" infile >outfile          ~( UnxUtils sed v4.0.7 or higher)
tr -d \r <infile >outfile             ~( GNU tr version 1.22 or higher )
```

Delete leading whitespace (spaces, tabs) from front of each line Aligns all text flush left

```
sed 's/^[ \t]*//'                     ~( see note on '\t' at end of file
  ⇒  )
```

Delete trailing whitespace (spaces, tabs) from end of each line

```
sed 's/[ \t]*$//'                    ~( see note on '\t' at end of file
  ⇒  )
```

Delete BOTH leading and trailing whitespace from each line

```
sed 's/^[ \t]*//;s/[ \t]*$//'
```

Insert 5 blank spaces at beginning of each line (make page offset)

```
sed 's/^/     /'
```

Align all text flush right on a 79-column width

```
sed -e :a -e 's/^.\{1,78\}$/ &/;ta'   ~( set at 78 plus 1 space )
```

Center all text in the middle of 79-column width. In method 1, Spaces at the beginning of the line are significant, and trailing Spaces are appended at the end of the line. In method 2, spaces at The beginning of the line are discarded in centering the line, and No trailing spaces appear at the end of lines.

```
sed  -e :a -e 's/^.\{1,77\}$/ & /;ta'                    ~( method 1 )
```

```
sed  -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/\( *\)\1/\1/'   ~( method 2 )
```

Substitute (find and replace) "foo" with "bar" on each line

```
sed 's/foo/bar/'              ~( replaces only 1st instance in a line )
```

```
sed 's/foo/bar/4'             ~( replaces only 4th instance in a line )
```

```
sed 's/foo/bar/g'             ~( replaces ALL instances in a line )
```

```
sed 's/\(.*\)foo\(.*foo\)/\1bar\2/'  ~( replace the next-to-last case )
```

```
sed 's/\(.*\)foo/\1bar/'              ~( replace only the last case )
```

Substitute "foo" with "bar" ONLY for lines which contain "baz"

```
sed '/baz/s/foo/bar/g'
```

Substitute "foo" with "bar" EXCEPT for lines which contain "baz"

```
sed '/baz/!s/foo/bar/g'
```

Change "scarlet" or "ruby" or "puce" to "red"

```
sed 's/scarlet/red/g;s/ruby/red/g;s/puce/red/g'    ~( most seds )
```

```
gsed 's/scarlet\|ruby\|puce/red/g'                  ~( GNU sed only )
```

Reverse order of lines (emulates "tac") Bug/feature in HHsed v1.5 causes blank lines to be deleted

```
sed '1!G;h;$!d'                ~( method 1 )
```

```
sed -n '1!G;h;$p'              ~( method 2 )
```

Reverse each character on the line (emulates "rev")

```
sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'
```

Join pairs of lines side-by-side (like "paste")

```
sed '$!N;s/\n/ /'
```

If a line ends with a backslash, append the next line to it

```
sed -e :a -e '/\\$/N; s/\\\n//; ta'
```

If a line begins with an equal sign, append it to the previous line And replace the "=" with a single space

```
sed -e :a -e '$!N;s/\n=/ /;ta' -e 'P;D'
```

Add commas to numeric strings, changing "1234567" to "1,234,567"

```
gsed ':a;s/\B[0-9]\{3\}\>/,&/;ta'                    ~( GNU sed )
```

```
sed -e :a -e 's/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta'   ~( other seds )
```

Add commas to numbers with decimal points and minus signs (GNU sed)

```
gsed -r ':a;s/(^|[^0-9.])([0-9]+)([0-9]{3})/\1\2,\3/g;ta'
```

Add a blank line every 5 lines (after lines 5, 10, 15, 20, etc.)

```
gsed '0~5G'                    ~( GNU sed only )
```

```
sed 'n;n;n;n;G;'               ~( other seds )
```

Section 25

Selective Printing Of Certain Lines

Print first 10 lines of file (emulates behavior of "head")

```
sed 10q
```

Print first line of file (emulates "head -1")

```
sed q
```

Print the last 10 lines of a file (emulates "tail")

```
sed -e :a -e '$q;N;11,$D;ba'
```

Print the last 2 lines of a file (emulates "tail -2")

```
sed '$!N;$!D'
```

Print the last line of a file (emulates "tail -1")

```
sed '$!d'                      ~( method 1 )
```

```
sed -n '$p'                    ~( method 2 )
```

Print the next-to-the-last line of a file

```
sed -e '$!{h;d;}' -e x              ~( for 1-line files, print blank
  ⇒ line )
```

```
sed -e '1{$q;}' -e '$!{h;d;}' -e x  ~( for 1-line files, print the line
  ⇒  )
```

```
sed -e '1{$d;}' -e '$!{h;d;}' -e x  ~( for 1-line files, print nothing
  ⇒ )
```

Print only lines which match regular expression (emulates "grep")

```
sed -n '/regexp/p'             ~( method 1 )
```

```
sed '/regexp/!d'               ~( method 2 )
```

Print only lines which do NOT match regexp (emulates "grep -v")

```
sed -n '/regexp/!p'            ~( method 1, corresponds to above )
```

```
sed '/regexp/d'                ~( method 2, simpler syntax )
```

Print the line immediately before a regexp, but not the line Containing the regexp

```
sed -n '/regexp/{g;1!p;};h'
```

Print the line immediately after a regexp, but not the line Containing the regexp

```
sed -n '/regexp/{n;p;}'
```

Print 1 line of context before and after regexp, with line number Indicating where the regexp occurred (similar to "grep -A1 -B1")

```
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h
```

Grep for AAA and BBB and CCC (in any order)

```
sed '/AAA/!d; /BBB/!d; /CCC/!d'
```

Grep for AAA and BBB and CCC (in that order)

```
sed '/AAA.*BBB.*CCC/!d'
```

Grep for AAA or BBB or CCC (emulates "egrep")

```
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d     ~( most seds )
```

```
gsed '/AAA\|BBB\|CCC/!d'                          ~( GNU sed only )
```

Print paragraph if it contains AAA (blank lines separate paragraphs) HHsed v1.5 must insert a 'G;' after 'x;' in the next 3 scripts below

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'
```

Print paragraph if it contains AAA and BBB and CCC (in any order)

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;/BBB/!d;/CCC/!d'
```

Print paragraph if it contains AAA or BBB or CCC

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
```

```
gsed '/./{H;$!d;};x;/AAA\|BBB\|CCC/b;d'           ~( GNU sed only )
```

Print only lines of 65 characters or longer

```
sed -n '/^.\{65\}/p'
```

Print only lines of less than 65 characters

```
sed -n '/^.\{65\}/!p'          ~( method 1, corresponds to above )
```

```
sed '/^.\{65\}/d'              ~( method 2, simpler syntax )
```

Print section of file from regular expression to end of file

```
sed -n '/regexp/,$p'
```

Print section of file based on line numbers (lines 8-12, inclusive)

```
sed -n '8,12p'
```

```
sed '8,12!d'                ~(the same)
```

Print line number 52

```
sed -n '52p'
```

```
sed '52!d'                  ~(another way to do it )
```

```
sed '52q;d'                 ~(another way, quicker for large files )
```

Beginning at line 3, print every 7th line

```
gsed -n '3~7p'                  ~( GNU sed only )
```

```
sed -n '3,${p;n;n;n;n;n;n;}' ~( other seds )
```

Print section of file between two regular expressions (inclusive)

```
sed -n '/Iowa/,/Montana/p'              ~( case sensitive )
```

Selective Deletion Of Certain Lines

Print all of file EXCEPT section between 2 regular expressions

```
sed '/Iowa/,/Montana/d'
```

Delete duplicate, consecutive lines from a file (emulates "uniq"). First line in a set of duplicate lines is kept, rest are deleted.

```
sed '$!N; /^\(.*\)\n\1$/!P; D'
```

Delete duplicate, nonconsecutive lines from a file. Beware not to Overflow the buffer size of the hold space, or else use GNU sed.

```
sed -n 'G; s/\n/&&/; /^\([ -~]*\n\).*\n\1/d; s/\n//; h; P'
```

Delete all lines except duplicate lines (emulates "uniq -d").

```
sed '$!N; s/^\(.*\)\n\1$/\1/; t; D'
```

Delete the first 10 lines of a file

```
sed '1,10d'
```

Delete the last line of a file

```
sed '$d'
```

Delete the last 2 lines of a file

```
sed 'N;$!P;$!D;$d'
```

Delete the last 10 lines of a file

```
sed -e :a -e '$d;N;2,10ba' -e 'P;D'     ~( method 1 )
```

```
sed -n -e :a -e '1,10!{P;N;D;};N;ba'    ~( method 2 )
```

Delete every 8th line

```
gsed '0~8d'                             ~( GNU sed only )
```

```
sed 'n;n;n;n;n;n;n;d;'                  ~( other seds )
```

Delete lines matching pattern

```
sed '/pattern/d'
```

Delete ALL blank lines from a file (same as "grep '.' ")

```
sed '/^$/d'                             ~( method 1 )
```

```
sed '/./!d'                             ~( method 2 )
```

Delete all CONSECUTIVE blank lines from file except the first; also Deletes all blank lines from top and end of file (emulates "cat -s")

```
sed '/./,/^$/!d'            ~( method 1, allows 0 blanks at top, 1 at EOF
  ⇒  )
```

```
sed '/^$/N;/\n$/D'          ~( method 2, allows 1 blank at top, 0 at EOF
  ⇒ )
```

Delete all CONSECUTIVE blank lines from file except the first 2

```
sed '/^$/N;/\n$/N;//D'
```

Delete all leading blank lines at top of file

```
sed '/./,$!d'
```

Delete all trailing blank lines at end of file

```
sed -e :a -e '/^\n*$/{$d;N;ba' -e '}'   ~( works on all seds )
```

```
sed -e :a -e '/^\n*$/N;/\n$/ba'         ~( ditto, except for gsed 3.02.*
  ⇒  )
```

Delete the last line of each paragraph

```
sed -n '/^$/{p;h;};/./{x;/./p;}'
```

27.1 Unix Man Pages

Remove nroff overstrikes (char, backspace) from man pages. The 'echo' Command may need an -e switch if you use Unix System V or bash shell.

```
sed "s/.`echo \\\b`//g"  ~(double quotes needed for Unix environment)
sed 's/.^H//g'           ~(in bash/tcsh, press Ctrl-V and then Ctrl-H)
sed 's/.\x08//g'         ~(hex expression for sed 1.5, GNU sed, ssed)
```

27.2 Email Messages

Get Usenet/e-mail message header

```
sed '/^$/q'              ~( deletes everything after first blank line )
```

Get Usenet/e-mail message body

```
sed '1,/^$/d'            ~( deletes everything up to first blank line )
```

Get the subject header, but remove initial "Subject: " portion

```
sed '/^Subject: */!d; s///;q'
```

Extract an email return address header

```
sed '/^Reply-To:/q; /^From:/h; /./d;g;q'
```

Parse out the address proper. Pulls out the e-mail address by itself From the 1-line return address header (see preceding script)

```
sed 's/ *(.*)//; s/>.*//; s/.*[:<] *//'
```

Add a leading angle bracket and space to each line (quote a message)

```
sed 's/^/> /'
```

Delete leading angle bracket & space from each line (unquote a message)

```
sed 's/^> //'
```

Remove most HTML tags (accommodates multiple-line tags)

```
sed -e :a -e 's/<[^>]*>//g;/</N;//ba'
```

Extract multi-part uuencoded binaries, removing extraneous header Info, so that only the uuencoded portion remains. Files passed to Sed must be passed in the proper order. Version 1 can be entered From the command line; version 2 can be made into an executable Unix shell script. (Modified from a script by Rahul Dhesi.)

```
sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode    ~( vers. 1 )
sed '/^end/,/^begin/d' "$@" | uudecode                     ~( vers. 2 )
```

Sort paragraphs of file alphabetically. Paragraphs are separated by blank Lines. GNU sed uses \v for vertical tab, or any unique char will do.

```
sed '/./{H;d;};x;s/\n/={NL}=/g' file | sort | sed '1s/={NL}=//;s/={NL
  ⇒ }=/\n/g'
gsed '/./{H;d};x;y/\n/\v/' file | sort | sed '1s/\v//;y/\v/\n/'
```

Zip up each .TXT file individually, deleting the source file and Setting the name of each .ZIP file to the basename of the .TXT file (under DOS: the "dir /b" switch returns bare filenames in all caps). echo @echo off >zipup.bat dir /b *.txt — sed "s/^\(.*\)\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat

Section 28

Quoting Syntax

The preceding examples use single quotes ('...') instead of double quotes ("...") to enclose editing commands, since sed is typically used on a Unix platform. Single quotes prevent the Unix shell from intrepreting the dollar sign ($) and backquotes (`...`), which are expanded by the shell if they are enclosed in double quotes. Users of the "csh" shell and derivatives will also need to quote the exclamation mark (!) with the backslash (i.e., \!) to properly run the examples listed above, even within single quotes. Versions of

```
sed written for DOS invariably require double quotes ("...") instead of
  ⇒  single
```

quotes to enclose editing commands.

Section 29

Use Of Tab In Sed Scripts

This section deals with the use of the \t sequence in sed scripts For clarity in documentation, we have used the expression '\t' to indicate a tab character (0x09) in the scripts. However, most versions of sed do not recognize the '\t' abbreviation, so when typing these scripts from the command line, you should press the TAB key instead. '\t' is supported as a regular expression metacharacter in awk, perl, and HHsed, sedmod, and GNU sed v3.02.80.

Section 30

Versions Of Sed

Versions of sed do differ, and some slight syntax variation is to be expected. In particular, most do not support the use of labels (:name) or branch instructions (b,t) within editing commands, except at the end of those commands. We have used the syntax which will be portable to most users of sed, even though the popular GNU versions of sed allow a more succinct syntax. When the reader sees a fairly long command such as this

```
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
```

it is heartening to know that GNU sed will let you reduce it to:

```
sed '/AAA/b;/BBB/b;/CCC/b;d'        ~( or even )
```

```
sed '/AAA\|BBB\|CCC/b;d'
```

In addition, remember that while many versions of sed accept a command like "/one/ s/RE1/RE2/", some do NOT allow "/one/! s/RE1/RE2/", which contains space before the 's'. Omit the space when typing the command.

Section 31

Optimizing For Speed

If execution speed needs to be increased (due to large input files or slow processors or hard disks), substitution will be executed more quickly if the "find" expression is specified before giving the "s/.../.../" instruction. Thus:

```
sed 's/foo/bar/g' filename          ~( standard replace command )
```

```
sed '/foo/ s/foo/bar/g' filename    ~( executes more quickly )
```

```
sed '/foo/ s//bar/g' filename       ~( shorthand sed syntax )
```

On line selection or deletion in which you only need to output lines from the first part of the file, a "quit" command (q) in the script will drastically reduce processing time for large files. Thus:

```
sed -n '45,50p' filename            ~( print line nos. 45-50 of a file )
```

```
sed -n '51q;45,50p' filename        ~( same, but executes much faster )
```

Sed one line contributors

Al Aab	Founder of "seders" list
Edgar Allen	Various
Yiorgos Adamopoulos	Various
Dale Dougherty	Author of "sed & awk"
Carlos Duarte	Author of "do it with sed"
Eric Pement	Author of this document
Ken Pizzini	Author of GNU sed v3.02
S.G. Ravenhall	Great de-html script
Greg Ubben	Many contributions & much help