**Handy one-liners for SED**

HANDY ONE-LINERS FOR SED (Unix stream editor) Mar. 23, 2001

compiled by Eric Pement <pemente@northpark.edu> version 5.1

Latest version of this file is usually at:

http://www.student.northpark.edu/pemente/sed/sed1line.txt

http://www.cornerstonemag.com/sed/sed1line.txt

This file is also available in Portuguese at:

http://www.lrv.ufsc.br/wmaker/sed\_ptBR.html

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'

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 '$='

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$//' # gsed 3.02.80, but top script is easier

# 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

# 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

# Cannot be done with DOS versions of sed. Use "tr" instead.

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 ':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

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 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' # method 1

sed '8,12!d' # method 2

# print line number 52

sed -n '52p' # method 1

sed '52!d' # method 2

sed '52q;d' # method 3, efficient on 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 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 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;}'

SPECIAL APPLICATIONS:

# 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 required 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 v1.5

# 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 Subject header, but remove initial "Subject: " portion

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

# get 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

# 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

TYPICAL USE: 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

For additional syntax instructions, including the way to apply editing

commands from a disk file instead of the command line, consult "sed &

awk, 2nd Edition," by Dale Dougherty and Arnold Robbins (O'Reilly,

1997; http://www.ora.com), "UNIX Text Processing," by Dale Dougherty

and Tim O'Reilly (Hayden Books, 1987) or the tutorials by Mike Arst

distributed in U-SEDIT2.ZIP (many sites). To fully exploit the power

of sed, one must understand "regular expressions." For this, see

"Mastering Regular Expressions" by Jeffrey Friedl (O'Reilly, 1997).

The manual ("man") pages on Unix systems may be helpful (try "man

sed", "man regexp", or the subsection on regular expressions in "man

ed"), but man pages are notoriously difficult. They are not written to

teach sed use or regexps to first-time users, but as a reference text

for those already acquainted with these tools.

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.

USE OF '\t' 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.

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.

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

If you have any additional scripts to contribute or if you find errors

in this document, please send e-mail to the compiler. Indicate the

version of sed you used, the operating system it was compiled for, and

the nature of the problem. Various scripts in this file were written

or contributed by:

Al Aab <af137@freenet.toronto.on.ca> # "seders" list moderator

Edgar Allen <era@sky.net> # various

Yiorgos Adamopoulos <adamo@softlab.ece.ntua.gr>

Dale Dougherty <dale@songline.com> # author of "sed & awk"

Carlos Duarte <cdua@algos.inesc.pt> # author of "do it with sed"

Eric Pement <pemente@northpark.edu> # author of this document

Ken Pizzini <ken@halcyon.com> # author of GNU sed v3.02

S.G. Ravenhall <stew.ravenhall@totalise.co.uk> # great de-html script

Greg Ubben <gsu@romulus.ncsc.mil> # many contributions & much help

*tpl@eng.cam.ac.uk*