Cheatography

Sed Cheat Sheet

by [deleted] via cheatography.com/30394/cs/9098/

regexp	
wild characters	
char	a single character, if not special, is matched against text
.	matches any character
*	matches a sequence of 0 or more repetitions of previous character/grouped regexp/class
.*	match all characters on every line (including empty ones)
\+	as *, but matches 1 or more
.\+	match all characters on every non-empty line
\?	as *, but only matches 0 or 1 character
special characters	
\n	new line
\t	tab
\s	whitespace
\S	any non-whitespace character
\w	any word character (letter, digit, underscore)
\W	any non-word character
line beginning	
^	matches the null string at the beginning of the line. What appears after ^ must appear at the beginning of the line
^#	match every line beginning with a # character
line ending	
$	same as ^, but refers to the end of line
\$	dollar sign is escaped, so this matches lines ending with a single dollar
\\$	backslash is escaped, so this matches lines ending with a single backslash
number of sequences	
\{*i*\}	as *, but matches exactly *i* number of sequences
\{i,\}	matches more than or equal to *i* sequences
\{i,j\}	matches between *i* and *j* sequences, inclusive
.\{9\}A$	matches an *A* that is the last character on line, with at least 9 preceding characters
^.\{15\}A	matches an *A* that is the 16th character on a line
groups and lists	
[*list*]	matches any single character in list. Dashes indicate inclusive sequences.
[a-zA-Z0-9]	matches any letters or digits

Sed Cheat Sheet

by [deleted] via cheatography.com/30394/cs/9098/

printing	
sed '' *file*	auto print file contents to command line (essentially `cat`)
sed -n 'p' *file*	`-n` surpresses auto printing of each line; `p` prints each line (same result as above)
sed -n '1p' *file*	print only 1st line
sed -n '1,5p' *file*	print 1st through 5th lines
sed -n '1,+4p' *file*	print 1st line and next 4 lines (same output as above)
sed -n '1~2p' *file*	print every 2nd line beginning with the 1st
sed -n '/keyword/p' *file*	prints every line that contains *keyword*

deleting	
sed '1~2d' *file*	delete every 2nd line beginning with the 1st (without `-n` option will also print what remains)
sed '/^$/d' *file*	matches any blank lines and passes them to the delete command
sed '/^$/!d' *file*	delete any line that is not blank (! inverts the address)
sed 's/[0-9]//g' *file*	delete all digits in all lines

interact with files	
sed '1~2d' *file* > *newfile*	delete every 2nd line from *file*, print remaining lines to *newfile*
sed -i '1~2d' *file*	delete every 2nd line "in-place", changes original file
sed -i.bak '1~2d' *file*	create backup file with .bak extension, edit the regular file in place

Note that source file is not affected by the basic commands; the edits are directed only to the command line unless explicitly directed to a file.

substitute	
basic substitution	
sed 's/oldword/newword/'	change 1st instance of *oldword* in each line to *newword*
sed 's|dir1/oldword|dir2/newword|'	if string includes forward slash, other valid delimiters include pipe, underscore
start options	
sed 'keyword/s/oldword/newword/' *file*	replace first instance of *oldword* with *newword* in any line that includes *keyword*
sed '1,3s/.*/newword/' *file*	substitute everything in 1st, 2nd, and 3rd lines with *newword*
end flags	
sed 's/oldword/newword/*flag*' *file*	*flag* can be any of the below
g	substitute every instance of oldword instead of just the first on each line (default behavior)

substitute (cont)	
2	substitute only *number*th instance of oldword on each line
p	print new pattern space for all lines where substitution was made
i	ignore case
other commands	
sed 's/word/(&)/' *file*	`&` holds matched pattern (*word*) and puts parentheses around it
sed 's/old/new/';s/first/second/' *file*	semicolon strings together distinct commands
examples	
sed -n 's/oldword/newword/2p' *file*	prints the lines where substitution took place

random tricks	
sed 's/.*/\"&\"/g'	add double quotes to line
sed "s|.*|$dirname|" *file*	use double quotes to expand variables within replacement. Use a different delimiter if variable contains slashes (e.g. directory path)
var=$(sed -n "${i}p" *file*)	set variable equal to line in *file*, where line number *i* is also a variable

Also remember `sed` reads and operates line by line. Some commands modify the output stream directly (so can't use results for more editing unless pipe it to another `sed` command).