

regex

wild characters

<i>char</i>	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>i</i> number of sequences
\{i,\}	matches more than or equal to <i>i</i> sequences
\{i,j\}	matches between <i>i</i> and <i>j</i> sequences, inclusive
.\{9\}A\$	matches an <i>A</i> that is the last character on line, with at least 9 preceding characters
^\{15\}A	matches an <i>A</i> that is the 16th character on a line

groups and lists

[i s f]	matches any single character in list. Dashes indicate inclusive sequences.
[a-zA-Z0-9]	matches any letters or digits



printing

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

deleting

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

interact with files

<code>sed '1~2d' file > newfile</code>	delete every 2nd line from <i>file</i> , print remaining lines to <i>newfile</i>
<code>sed -i '1~2d' file</code>	delete every 2nd line "in-place", changes original file
<code>sed -i.bak '1~2d' file</code>	create backup file with <code>.bak</code> 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

<code>sed 's/oldword/newword/'</code>	change 1st instance of <i>oldword</i> in each line to <i>newword</i>
<code>sed 's dir1/oldword dir2/newword '</code>	if string includes forward slash, other valid delimiters include pipe, underscore

start options

<code>sed 'keyword/s/oldword/newword/' file</code>	replace first instance of <i>oldword</i> with <i>newword</i> in any line that includes <i>keyword</i>
<code>sed '1,3s././newword/' file</code>	substitute everything in 1st, 2nd, and 3rd lines with <i>newword</i>

end flags

<code>sed 's/oldword/newword/flag' file</code>	<i>flag</i> can be any of the below
<code>g</code>	substitute every instance of <i>oldword</i> instead of just the first on each line (default behavior)

C

By [deleted]
cheatography.com/deleted-30394/

Not published yet.
 Last updated 3rd October, 2016.
 Page 2 of 3.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopod.com>

substitute (cont)

2	substitute only <i>numberth</i> instance of oldword on each line
p	print new pattern space for all lines where substitution was made
i	ignore case

other commands

<code>sed 's/word/(&)' file</code>	& holds matched pattern (<i>word</i>) and puts parentheses around it
<code>sed 's/old/new/;s/first/second/' file</code>	semicolon strings together distinct commands

examples

<code>sed -n 's/oldword/newword/2p' file</code>	prints the lines where substitution took place
---	--

random tricks

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



By [deleted]

cheatography.com/deleted-30394/

Not published yet.

Last updated 3rd October, 2016.

Page 3 of 3.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>