**Stream Editing with sed**

If you want to write a program to make changes in a file, sed is the tool to use. Sed is the ultimate **s**tream **ed**itor.

In general, the usage is:

sed [options] commands [file-to-edit]

The most commonly used command is the substitute command “s”. For example:

echo day | sed s/day/night/

This outputs “night” because it substitutes “night” for “day”.

echo Sunday | sed s/day/night/

Sunnight

What if we don’t want that behavior? We could try adding a space:

echo Sunday | sed s/ day/night/

If the command contains meta-characters (including spaces) you need quotes.

echo Sunday | ‘sed s/ day/night/’

It’s best practice to always use quotes.

Let’s make a text file called “example.txt” for our next example:

apple banana kiwi burger

orange kiwi pie burger pie berry

sed ‘s/banana/mango/’ <example.txt >test.txt

This will create “test.txt” which will read:

apple mango kiwi burger

orange kiwi pie burger pie berry

Let’s try something else. Sed is line oriented, so

sed ‘s/kiwi/wiki/’ <example.txt >test.txt

will yield:

apple banana wiki burger

orange wiki pie burger pie berry

As we would expect, it changes all instances of “kiwi”.

However,

sed ‘s/pie/cobbler/’ <example.txt >test.txt

will yield:

apple banana kiwi burger

orange kiwi cobbler burger pie berry

Because sed only substitutes one instance per line by default.

To change this, we can pass in an option specifying to substitute **g**lobally:

sed ‘s/pie/cobbler/g’ <example.txt >test.txt

will yield:

apple banana kiwi burger

orange kiwi cobbler burger cobler berry

There are other flags too.

<https://www.gnu.org/software/sed/manual/html_node/The-_0022s_0022-Command.html>

You can also use regular expressions to match strings. Let’s change our example file to:

kawi kewi kiwi kowi kuwi kywi

Then we can do:

sed ‘s/k.wi/kiwi/g’ <example.txt >test.txt

To get:

kiwi kiwi kiwi kiwi kiwi kiwi

You can use the “&” character as the string found. Let’s say we want to put parentheses around all of the words in the file. We need to use the string found in the replacement string:

sed ‘s/k.wi/(&)/g’ <example.txt >test.txt

This gives:

(kawi) (kewi) (kiwi) (kowi) (kuwi) (kywi)

If you have a large number of sed commands, you can put them into a file and use:

sed -f sedscript <old >new

where sedscript could look like this:

# This script changes lower case vowels to upper case

s/a/A/g

s/e/E/g

s/i/I/g

s/o/O/g

s/u/U/g

When there are several commands in one file, each command must be on a separate line.

Use # for comments.

In addition to “s” for substitution, we can use “p” to print.

To print the first line:

sed –n ‘1p’ example.txt

The –n option suppresses the default automatic printing behavior of sed.

We can print every other line:

sed -n '1~2p' BSD

We can also use sed to delete text with the “d” command.

To delete every other line:

sed '1~2d' example.txt

To save this output:

sed '1~2d' example.txt > test.txt

See for more:

<https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-editor-to-manipulate-text-in-linux>

Here is a list of useful sed one-liners:

<http://sed.sourceforge.net/sed1line.txt>

Here’s another good blog post that I used for this:

<http://www.grymoire.com/Unix/Sed.html>