sed Command

coursera.org/learn/linux-tools-for-developers/supplement/ryYmz/sed-command

sed stands for stream editor. Its job is to make substitutions and other modifications in files and in streamed output.

Any of the following methods will change all first instances of the string "pig" with "cow" for each line of **file**, and put the results in **newfile**:

```
$ sed s/pig/cow/ file > newfile

$ sed s/pig/cow/ < file > newfile

$ cat file | sed s/pig/cow/ > newfile
```

where the **s** stands for substitute. If you want to change all instances, you have to add the **g** (global) qualifier, as in:

```
$ sed s/pig/cow/g file > newfile
```

The / characters are used to delimit the new and old strings. You can choose to use another character, as in:

```
$ sed s:pig:cow:g file > newfile
```

Some of the complications come in when you want to use special characters in the strings to be searched for or inserted. For example, suppose you want to replace all back slashes with forward slashes:

```
$ sed s/’\\’/’\/’/g file > newfile
```

It is never a bad idea to put the strings in either single or double quotes, the main difference being that environment variables and escaped special characters in double quotes are expanded out while they are not in single quotes, as in:

```
$ echo "$HOME"

/home/coop

$ echo ’$HOME’

$HOME
```

If you want to make multiple simultaneous substitutions, you need to use the **-e** option, as in:

```
$ sed -e s/"pig"/"cow"/g -e s/"dog"/"cat"/g < file > newfile
```

and you can work directly on streams generated from commands, as in:

```
$ echo hello | sed s/"hello"/"goodbye"/g

goodbye
```

If you have a lot of commands, you can put them in a file and apply the **-f** option, as in:

```
$ cat scriptfile

s/pig/cow/g

s/dog/cat/g

s/frog/toad/g

$ sed -f scriptfile < file > newfile
```