sed

sed is short for **s**tream **ed**itor. It reads in a file (one line at a time), modifies the contents per your parameters, and then outputs.

A more simple description, it allows you to modify the contents of a file without you actually having to open it.

Basics

Sample File

Our sample text file called **hey.txt** looks like this,

```
Hey, diddle, diddle,
The cat and the fiddle,
The cow jumped over the moon;
The little dog laughed
To see such sport,
And the dish ran away with the spoon.

Make sure to keep this file in your /opt/nursery/ directory.
```

Create this file with the above contents and also create a pristine version. We'll keep on reseting the file by copying back from the pristine version for each sample.

Simple Replace

Here a simple example of replacing the word cow with reindeer in the file **hey.txt,**

```
sed -i 's/cow/reindeer/' hey.txt
```

The -i signifies inline editing.

If you would like sed to automatically create a backup file specify the suffix of the backup file by adding .suffix. For example to create the backup file with the extension **.sedautobck**,

```
sed -i.sedautobck 's/cow/reindeer/' hey.txt
```

Reserved Character

Here is an often used example of escaping a the / (slash) which is a reserved character within the single quote,

```
sed -i 's/opt\/nursery/var\/ryhme/' hey.txt # need to escape '/' since we
are using it as a separator
```

However, sed permits **any** character other than backslash or newline to be used instead of the forward slash. For example, the above example may be rewritten as follows to avoid needing to escape the forward slashes (sometimes referred to as "picket fences"):

```
sed -i 's,original/path,new/path,' hey.txt # do not need to escape '/'
since we are using ',' as a separator
```

Detect if No Match Found

sed will exit gracefully with a return code of 0 even if a match is found. The sed return code by default is based on running properly not if your data matches.

One of the problems I've run into, thinking I successfully modified my files even though I did not. Here, I've made a mistake of using the word **coy** instead of **cow**,

```
sed -i.sedautobck 's/coy/reindeer/' hey.txt
cmp -s hey.txt hey.txt.sedautobck && echo "sed did not work, your files are
identical."
sed did not work, your files are identical.
```

With the mistake you get the extra third line, "**sed did not work, your files are identical.**".

And if we want to be more fancy to make the cmp line more generic,

```
sed -i.sedautobck 's/coy/reindeer/' hey.txt
cmp -s $_ $_.sedautobck && echo "sed did not work, your files are
identical."
```

Insert Multiple Lines from File with Match

..

```
sed '/cdef/r muliple-lines.txt' hey.txt
```

..

References

Simple introduction - http://www.grymoire.com/Unix/Sed.html
Another introduction - http://lowfatlinux.com/linux-sed.html

Using sed to edit and save to the same file - http://unstableme.blogspot.ca/2010/01/sed-save-changes-to-same-file.html