Using perl and sed to substitute strings in multiple files on Linux and BSD

Author : admin

Perl and Sed replace strings tutor on Linux and BSD

On many occasions when had to administer on Linux, BSD, SunOS or any other *nix, there is a need to substitute strings inside files or group of files containing a certain string with another one.

The task is not too complex and many of the senior sysadmins out there would certainly already has faced this requirement and probably had a good idea on files substitution with perl and sed, however I'm quite sure there are dozen of system administrators out there who did not know, how and still haven't faced a situation where there i a requirement to substitute from a command shell or via a scripting language.

This article tagets exactly these system administrators who are not 100% sys op Gurus ;)

1. Substitute text strings inside files on Linux and BSD with perl

Perl programming language has originally been created to do a lot of text manipulation as well as most of the Linux / Unix based hosts today have installed working copy of **perl** , therefore using *perl* as a mean to substitute one string in a file to another one is maybe the best way to completet the task.
Another good thing about **perl** is that text processing with it is said to be in most cases a bit faster than **sed** .
However it is still dependent on the string to be substituted I haven't done benchmark tests to positively say 100% that always perl is quicker, however my common sense suggests perl will be quicker.

Now enough talk here is a very simple way to substitute a reoccuring, text string inside a file with another chosen one is like so:

```
debian:~# perl -pi -e 's/foo/bar/g' file1 file2
```

This will substitute the string **foo** with **bar** everywhere it's matched in **file1** and **file2**

However the above code is a bit "dangerous" as it does not preserve a backup copy of the original files, where string is substituted is not made.
Therefore using the above command should only be used where one is 100% sure about the string changes to be made.

Hence a better idea whether conducting the text substitution is to keep also the original file backup under a let's say **.bak** extension. To achieve that I use perl as follows:

```
freebsd# perl -i.bak -p -e 's/syzdarma/magdanoz/g;' file1 file2
```

This command creates copies of the original files **file1** and **file2** under the names *file1.bak* and *file2.bak* , the files **file1** and **file2** text occurance of strings **syzdarma** will get substituted with **magdanoz** using the option */g* which means - (substitute globally).

2. Substitute string in all files inside directory using perl on Linux and BSD

Every now and then the there is a need to do manipulations with large amounts of files, I can't right now remember a good scenario where I had to change all occuring matching strings to anther one to all files located inside a directory, anyhow I've done this on a number of occasions.

A good way to do a **mass file string substitution** on Linux and BSD hosts equipped with a **bash shell** is via the commands:

```
debian:/root/textfiles:# for i in $(echo *.txt); do perl -i.bak -p -e 's/old_string/new_string/g;' $i; done
```

Where the text files had the default txt file extension **.txt**

Above bash loop prints each of the files located in **/root/textfiles** and substitutes everywhere (globally) the **old_string** with **new_string** .

Another alternative to the above example to **replace multiple occuring text string in all files in multiple directories** is possible using a combination of shell commands **grep, perl, sort, uniq and xargs** .
Let's say that one wants to **match everywhere inside the root directory and all the descendant directories for files with a custom string and substitute it to another one**, this can be done with the cmd:

```
debian:~# grep -R -files-with-matches 'old_string' / | sort | uniq | xargs perl -pi~ -e
'ss/old_string/new_string/g'
```

This command will lookup for string **old_string** in all files in the */ - root directory* and in case of occurance will substitute with **new_string** (This command's idea was borrowed as an idea from *http://linuxadmin.org* so thx.).

Using the combination of 5 commands, however is not very wise in terms of efficiency.

Therefore to save some system resources, its better in terms of efficiency to take advantage of the **find** command in combination with **xargs** , here is how:

```
debian:~# find / | xargs grep 'old_string' -sl |uniq | xargs perl -pi~ -e 's/old_string/new_string/g'
```

Once again the **find** command example will do exactly the same as the substitute method with **grep -R ...**

As enough is said about the way to substitute text strings inside files using *perl*, I will further explain how text strings can be substituted using **sed**

The main reason why using *sed* could be a better choice in some cases is that Unices are not equipped by default with perl interpreter. In general the amount of servers who contains installed **sed** compared to the ones with **perl** language interpreter is surely higher.

3. Substitute text strings inside files on Linux and BSD with *sed* stream editor

In many occasions, wether a website is hosted, one needs to quickly conduct a change in string inside all files located in a directory, to resolve issues with static urls directly encoded in **html**.
To achieve this task here is a code using two little bash script loops in conjunctions with sed, echo and mv commands:

```
debian:/var/www/website# for i in $(ls -1); do cat $i |sed -e
"s#index\.htm#http://www.webdomain.com/#g">$i.new; done
debian:/var/www/website# for i in $(ls *.new); do mv $i $(echo $i |sed -e "s#\.new##g"); done
```

The above command **sed -e "s#index\.htm#http://www.webdomain.com/#g"**, instructs sed to substitute all appearance of the text string **index.htm** to the new text string **http://www.webdomain.com**

First for bash loop, creates all the files with substituted string to *file1.new, file2.new, file3.new* etc.
The second for loop uses **mv** to overwrite the original input files *file1, file2, file3, etc.* with the newly created ones *file1.new, file2.new, file3.new*

There is a a way shorter way to conclude the same text substitutions task using a simpler one liner with only using sed and bash's eval capabilities, here is how:

```
debian:/var/www/website# sed -i 's/old_string/new_string/g' *
```

Above command will change **old_string** to **new_string** inside all files in directory **/var/www/website**

Whether a change has to be made with less than 1024 files using this method might be more efficient, however whether a text substitute has to be done to let's say **5000+** the above simplistic version will not work. An error of **Argument list too long** will prevent the *sed -i 's/old_string/new_string/g'* to complete its task.

The above **for** loop 2 liner should be also working without problems with FreeBSD and the rest of BSD derivatives, though I have not tested it yet, hence any feedback from FreeBSD guys is mostly welcome.

Consider that in order to have the **for** loops commands work on FreeBSD or NetBSD, they have to be run under a bash shell.
That's all folks thanks the Lord for letting me write this nice article, I hope it gives some insights on **how multiple files text replace on Unix works** .
Cheers ;)