**Sed: simple pattern address usage**

Most people I know use sed for simple and fast translation of some keyword in files. For instance, changing ports and tags inside configuration files during deployment to production servers. This results in sometimes clumsy scripts to make sure that sed changes a keword on line 4, but not on line 40. Most people I know have no idea that the way you can actually limit the range in which sed has to operate. Let’s explore…

Sed’s general form is something like this:

[address[,address]][!]command [args]

As you can see, it LOOKS  simple enough, but it really isn’t. Let’s take a look at a simple sed-action:

echo "Hello world" | sed -e 's/world/people/'

This will output “Hello people” since it changes the text “world” into “people”. If instead of 1 line, we streamed a file into sed, it will replace the first “world” string for every line in the file. If you have 2 “world” strings on one line, you need to add the “g” option (g meaning global here).

cat largefile.txt | sed -e 's/world/people/g'

**Meet sed’s addressing**

This is all fine and dandy, until you don’t want to replace all world-tags in a file, but just the tags in the first, say, 10 lines. It’s very easy to accomplish this in sed:

cat largefile.txt | sed -e '1,10 s/world/people/g'

This looks the same, only we have added an address-range to sed. In this case, it will change world into people on line 1 to 10, but sometimes we want to do the opposite: change on all lines EXCEPT 1 to 10. Still easy enough:

cat largefile.txt | sed -e '1,10 ! s/world/people/g'

**Now for the fun part**

It is not only possible to add line numbers for addressing, but you can also add a regex if you like (or any mix).

cat largefile.txt | sed -e '/^start/,/^end/ s/world/people/g'

this will change world to people for every line after finding a line that starts with “START”. It will stop after finding a line that starts with “END”. So let’s see, which lines will be changed in the following files:

hello world

this is a world readable test file

let's start shall we?

start

the world is not enough. But the world should be.

end

start with a better world

ending with a world of hate is not an option

Did you catch them all? Here’s the answer:

hello world

this is a world readable test file

let's start shall we?

start

the people is not enough. But the people should be.

end

start with a better people

ending with a people of hate is not an option

**An advanced example**

Let’s see how some more advanced usages work. These examples use application.ini, which is a zend framework configuration file, but basically any INI file will do. Let’s “remove” the [testing] section by commenting out the data. Note that we don’t know (nor care) where (or if) this [testing] section is present or how long it is.

cat application.ini |

sed -e '/^\[testing]/,/^\[/ { /^\[testing]/b ; /^\[/b; s/\(.\*\)/\*\*DELETED\*\* \1/ }'</pre>

So, what does this actually do then? Let’s take a look at the address range:

The address-range:

/^\[testing]/,/^\[/

this sed script matches (every) line that starts with [testing] and ends with a line that starts with [.  When dealing with ini-files, this basically means you match a complete [testing] section, INCLUDING the [testing] and next [..] lines. It looks a bit messy but the same /RE1,RE2/ syntax can be found.

The { } brackets makes it possible to group multiple sed commands just like php. In this case, we have 3 commands, all separated by a ; (sounds very familiar does it?)

/^\[testing]/b

This command tells sed to “branch” to the end of the script (and it will continue with a new line). In effect, it’s a check that no other commands will be run when we match the initial **[testing** block. It would be easier if sed had a way to exclude the address-ranges but alas. So we are forced to check the start and end matches ourselves.

/^\[/b

As said, we checked the start, so we have to check the end as well, so these lines don’t get changed by sed.

s/\(.\*\)/; \*\*DELETED\*\* \1/

When we arrived here, we can safely change our lines. In this case, it will do a substitution (the ‘s’) where it will match .\*, which is regex-speak for the whole line. The () are needed to capture this whole line into the \1 parameter we use inside the substition.

We substitute the line by changing the line to  “; \*\*DELETED\*\* “, and adding the \1 to it. This effectively appends the “; \*\* deleted \*\*” string in front of the line.

That’s it.. The whole INI-section called [testing] is commented out in one go and you know a bit more about sed’s address usage.

Enjoy..