**31. Data Manipulation and Text Transformation With SED**

- Learn most important features of SED commands

Sed = Stream Editor

* It is used for editing streams.
* We might be not familiar with streams, but we have been using it.
* We used in IO redirections and piping | we were using streams.
* We can consider stream as the data that travels from one process to another through a pipe or from one file to another as a redirect or from one device to another.
* We can take Stdin standard input as stream of input and stdout standard output as output stream and standard error as standard error stream.
* These data streams are typically texted data.
* The sed command is used to perform basic text transformation on an input stream.

For example we can use sed to substitute some text for other text, remove lines, a pen text after a given lines and insert text before certain lines.

* Sed is used programmatically not interactively like VIM or Nano.
* We can use or write macros to perform these functions in vim, but you must have someone to start vim and then execute the macros.
* With sed you can programmatically perform these edits without the need for interaction and this makes sed ideal to be used in shell script
* for example:

The most common use of SED is to act as a command line version of find and replace.

Sed is a standalone utility and not a shell built in.

Example for search and replace text in file using sed

$ cat test.txt

$ This is created for test

$ sed ‘S/This/This file/’ test.txt

$ This file is created for test

This does not edit the original file using this way but we can also edit and replace the desired text using sed utility

Syntax of sed search and replace with flags.

$ sed ‘s/search-pattern/replacement-string/flags’

* To match both uppercase and lowercase alphabets we need to use flags in sed commands as sed is case sensitive without flags.
* Using flags we can pass insensitive search pattern. For Insensitive search pattern we can use ‘i’ flag.
* Eg
  + $ sed ‘s/THIS/This file/i’ test.txt
  + the flag i make the command insensitive.
* All the above commands replace the first occurrences of any line of any file. It does not apply to multiple times in one line i.e only one time in one line,
* Again, here we can use flag to make the replacement globally.
* G can be considered as global replacement.
* Eg:
  + $ sed ‘s/THIS/This file/g’ test.txt
* To replace particular occurrence using sed we can use numbers as flags i.e
* Eg:
  + $ sed ‘s/THIS/This file/3’ test.txt
* This command will edit only the third occurrence in the line
* To edit and alter the search pattern in the original file use -i option in sed command
  + -i: place in editing
  + $ sed -i ‘s/THIS/This file/g’ test.txt
* If you want to make a backup copy os a file before editing the original file add some text right after the -i option and sed will append that text to the copy of a file
* Eg:
  + $ echo “I live my wife.” > test.txt
  + $ echo “This is line 2.” > test.txt
  + $ echo “I live my wife with all of my heart.” > test.txt
  + $ echo “I live my wife and my wife loves me. Also, my wife loves the cat.” > test.txt
  + $ sed -i.bak ‘s/THIS/This file/i’ test.txt
  + $ ls
  + life.txt life.txt.bak
* Note: Do not use space between -i option and backup name.
* If you only want to save the lines where matches were madel We can use the ‘w’ flag followed by a filename.
  + $ sed ‘s/love/like/gw like.txt’ test.txt
* While we are looking for input and output examples.
* Note that sed can be used in a pipeline instead specifying a file to work on.
* Let’s take car ad an example and pipe that to sed.
  + $ cat like.txt | sed ‘s/my wife/sed/g’
* Let’s generate some text using echo command
  + $ echo ‘/home/jason’
* If we want to use sed in replacing forward slash / we can do it by two approaches one is using the escape character backslash \.
* To make /home/Jason to /export/user/Jason
  + $ echo ‘/home/Jason’ | sed ‘s/\/home\/Jason/\/export\/users\/Jasonc/’
  + /export/users/Jasonc
* Another way to do this is using any character as the delimiter like (:, #, %)
* The first character after ‘s’ will be treated as a delimiter no matter what character that is.
* Eg:
  + $ echo ‘/home/Jason’ | sed ‘s#/home/Jason/#export/users/Jasonc/#’

We can also remove or delete search patterns using sed

* To delete using sed use ‘d’ command in sed.
* Eg
  + $ sed ‘/This/d’ test.txt
* This deletes the complete lines in which it contains ‘This’ in the line.
* Remove the line and that start s with #
* Let’s create some data first in a file.
  + $ echo “# User to tun service as” >> config
  + $ echo “User apache” >> config
  + $ echo >> config
  + $ echo “Group to run service as”>> config
  + $ echo “Group apache” >> config
  + $ sed ‘/^#/d’ config
* Search pattern is actually a Regular Expression.
* In a regular expression the caret (^) symbol matches only the beginning of the line, It matches position and not a character.
* This makes sure we don’t accidentally delete lines which had # in the middle of the lines

To remove blank lines

* To remove blank lines we use another regular expression
  + $ sed ‘/^$/d’ config
* Here
  + caret (^) represents: starting of a line and
  + dolar ($) represents: end of line.
* To do both actions simultaneously we can separate the commands using semi columns.
* Eg:
  + $ sed ‘/^#/d ; /^$/d’ config
* Deletion with replacement
  + $ sed ‘/^#/d ; /^$/d config ; s/apache/httpd’ config
* Another way to execute multiple sed commands.
* The reason to show this is just in case if we had to see it in another person’s script in which someone else have worked on we should know what’s going on.
* The other way to do this is to use -e option in ‘sed’
* -e: one for each command to execute
  + $ sed -e ‘/^#/d’ -e ‘/^$/d -e’ -e ‘s/apache/httpd’ config
* Sed also allows to specify a file containing the sed commands using -f option.
  + $ echo ‘/^#/d’ > script.sed
  + $ echo ‘/^$/d’ > script.sed
  + $ echo ‘s/apache/httpd/’ > script.sed
  + $ cat script.sed
  + /^#/d
  + /^$/d
  + s/apache/httpd/
  + sed -f script.sed config

Let’s take a look on addresses in sed

* A address determines on what line the sed command will be executed on.
* If no address is given the command is performed on all lines.
* An address is specified before the sed command.
* The simplest of address in a line number.
* To execute sed command on line 2 of a file
  + $ sed ‘2 s/apache/httpd’ config
* We can also use regular expression as a address to match lines
* Lets say we have to replace apache with httpd but only on line that contains Groups.
  + $ sed ‘/Group/ s/apache/httpd’ config
* We can also specify range separated with commas.
  + 1,4 = 1 to 4
* Eg:
  + Sed ‘1,5 s/run/execute/’ config
* We can also use regular expressions as range
* Eg:
  + $ sed /# User/, /^$/ s/run/execute’ config
* Now we get the point we could specify a large portion of the file if we wanted to.
* One starting point to another starting point as your address and then change everything else alone.
* It is very powerful to user addresses with sed.

**In this lesson**

* We have learned the most common use case of sed which is to perform text substitutions we learned how to replace specific occurrences as well as how to replace all occurrences of the search pattern.
* In addition to finding and replacing text we learned how to delete text with the d command.
* From there we have learned how to save the alterations performed by sed as well as how to make backups of the original file so that your date is safe.
* Next, we have learned there are different ways to execute multiple sed commands on the same set of data.
* Finally, we have learned how to use addresses to work on very specific sections of input.