# Day 2:

Mastering The Linux Command Line

***Different Text Editors in Linux***

* *Nano*
  + Gives a GUI Interface.
  + By default in editing mode
  + Ctrl + K = deletes the entire line
  + Ctrl + X gives small gui which asks if Yes, No or Cancel.
* *Vi* 
  + I – insert
  + :wq! – save and exit
  + :w! – only save
  + :q! – exit without saving
  + ESC – to come out of edit mode.

***Copying, Moving, & Removing Files & Directories***

* Whoami – will give user name
* Pwd = current working directory
* Mkdir = to create directory
* Ls = will list down all files and folders.
* Ls -al = will be listing down all detail listing of all items in working directory
* Files which start with ‘.’ are configuration files, which show up in detail listing only.
* ‘Man’ command is used to list all options for a specific command.
* Cd = enter into a directory.
* Mv = will rename file name [mv <old\_filename> <new\_filename>], same for directories
* Tail, head, cat = we can see the content in the file or logs
* Rm = remove the file from the directory [rm <filename>]
* go to home directory using ‘cd’ or ‘cd /home/user’ or ‘cd ~/’
* rmdir = remove directory but will work when directory is empty.
* Rm -r <dir name> = ‘-r’ is recursively remove files and directory.
* Touch = also creates a new file
* Cp = will copy old file content into new one. [cp <old\_filename> <new\_filename>]
* Cp -r = for directory.

***Command Line Keyboard Shortcuts***

* Ctrl + A = takes you start of line
* Ctrl + E = takes you to end of line
* Ctrl + K = clears content after the cursor including the cursor location.
* Ctrl + U = clears content before the cursor without including the cursor location.
* Ctrl + L = clears the screen with the current cursor location being seen.
* Ctrl + C = breaks out of current process.
* Ctrl + W = clears the last word.
* Ctrl + T = switches the character before the cursor into that cursor position
* Ctrl + H = delete the value before cursor
* Ctrl + D = delete the value after cursor
* Ctrl + P = takes to last command executed, which can be done by up and down arrow keys.

***Aliases***

* ‘Tail messages’, gives last 10 messages which is default.
* We can change number ‘tail -n 20 messages’ gives last 20.
* ‘head messages’, gives first 10 messages which is default.
* We can d -n 20 messages’ gives first 20.
* ‘tail -f messages’ will display the data as stdout format, when over a line gets added into messages then it will be displayed on the screen.
* Alias will create a separate name
* $ alias showlog = “tar -f /var/log/messages”
* Type showlog and you will be able to see the execution.
* We can get details of that alias by writing ‘alias showlog’ which will show which commands that showlog runs.
* If we exit from the system and login again then we cant get that command, for that we need to make changes in configuration file.

***Bash Shell Configuration File & Shell Variables***

* When we use ‘ls -al’ command then it lists all files, folders in detailing.
* If we see ‘.’ Files then those are config files, we need to change that ‘.bashrc’ or some times it might be incorporated in ‘.bashprofile’.
* If we add alias showlog = “tar -f /var/log/messages” in config file then it stays in the system even after logging out.
* Like wise we have shell variables, which would be defined by ‘export MYNAME= “MUKHASIR”’ on command line and using ‘echo $MYNAME’ we can see the value in that.
* We can set this in config file so that all users can use.
* If we use ‘env’ command then we can see all environment variables. If we do echo we can see those.

# Day 3:

***Setting and Unsetting Environment***

* Environment Variables contain configuration information for our specific shell.
* ‘SET’ command will list our all environment variables.
* ‘which <command(ls)>’ will give all details and its location.
* To echo value of variable put $ before the variable.
* ‘export’ command is used to set value in shell variable. [export PWD = “/home/whatever” will change root to whatever]
* The above command will get reset once we use another command or used CD.
* ‘mkdir -p /custom/log’ [-p = creates directory from the path if not present]
* For example:
  + export CUSTOM\_LOG = “/custom/log”
  + echo $CUSTOM\_LOG
  + echo “My log message” > $CUSTOM\_LOG/log.txt
  + export CUSTOM\_LOG = “/custom/log/log.txt”
  + echo “My log message” > $CUSTOM\_LOG [> = write in the file]
  + echo “My log message” >> $CUSTOM\_LOG [>> = append]
* ‘unset CUSTOM\_LOG’ will clear of the data in the variable, but will not delete variable.
* If we need it for all users we can add into ‘.bashprofile’ or got into ‘/etc’ and find for file/folders with profile, view profile file which says that its better to use custom.sh file to add any variables, so enter profile.d folder and then add custom.sh file.
* Grep is for search

***Customizing the Bash Prompt***

* Customize ‘PS’ varaiable.
* PS1 = ‘[\u@\h \W]\$’ (\u = root; \h =hostname; \W = end part of current working directory; )
* We can change the command prompt
* For Example
  + ‘\t’ will give time as prompt. (hh:mm:ss)
  + PS1 = ‘\t’
  + ‘\j’ will give number of jobs running in background
  + ‘\w’ (lowercase) will show entire working directory path.
  + ‘\d’ will show current date
  + ‘\H’ (capital) will give full hostname
  + ‘man bash’ will give all lsit of options that can be done.

***Command Line History***

* Bash history(.bash\_history) file save whatever we type on the command line.
* Using ‘history’ command we can get all list from command line history.
* To clear history ‘history -c’

***Creating a Simple bash script to automate commands***

* We write file, first line would be [shebang] ‘#!/bin/bash’ , this tells what interpreter is going to run my script.
* Script
  + #!/bin/bash
  + Echo “my script is running”
  + Ls /etc > ls-contents.txt
  + Cat ls-contents.txt | grep cron
  + Rm ls-contents.txt
* Now we can automate it using the cron job
* For script to be executed, need to add execute permission to file. [chmod +x <filename(myscript)>]
* Then ‘$ ./myscript’

**Finding Files in Linux**

***Which and Whereis***

* This commands will be useful to get information about other linux commands
* ‘which’ command allows us to see full path of shell commands and how it is created.
* ‘whereis’ command allows us to look at the binary, the source and the manual files for a specific command.
* ‘whereis pwd’
  + Pwd: /bin/pwd /usr/share/man/man1p/pwd.1p.gz /usr/share/man/man1/pwd.1.gz
  + 1St = Location
  + 2nd, 3rd = Where it is located in ‘man’ page. (would be part of man section 1)

***Finding Files with Locate***

* ‘locate’ command in order find files in linux.
* Locate will all the command names which have pwd in it. [locate pwd]
* Locate searched in the database rather in the complete system to make search faster.
* So we need to update db using ‘updatedb’ command, so that new items are added in.

***The Powerful Find Command***

* ‘find’ will do manipulate, find and more
* ‘find . -name ‘cron\*’’ [‘.’ = current working directory, ‘cron\*’ = searching for name starting with cron and ending with any thing]
* ‘find . -type f -name ‘cron\*’’ [-type f = would search only files] (list only files)
* ‘find . -type d -name ‘cron\*’’ [-type d =search only directory] (list only directories)
* Search for files in home directory which have ‘777’ permissions which means anyone can read, write and execute that file, it is a very insecure permission setting.
* ‘touch file\_name>’ would update the timestamp if file is present or would create new.
* ‘touch files(file1, file2, file3)’ and set “777” permissions using ‘chmod 777 <filename>’ and for other file set “555” permission.
* ‘find /home -perm 777’ will list all files with permission “777”.
* If we want to change permission for the resulted files then
  + ‘find /home -perm 777 -exec chmod 555 {} \;’ [added -exec to execute other command on the result set]
* Find all files modified in last one day, ‘find / -mtime +1’ (‘/’ = root folder, -mtime = modified time)
* Find all files accessed in last one day, ‘find / -a +1’ (‘/’ = root folder, -a = accessed time)
* Find with group name, ‘find / -group <nameofgroup>’
* Find all files with size of 512 bytes, ‘find /home -size 512c’ or ‘find . -size 512c’ (depends on the position on command line)
* Remove files from the folder, ‘find /home/user -name ‘file\*’ -exec rm {} \;’
* We can look for empty folder and delete them.

***Finding files in Linux Lab***

Yum -y update [update system packages in CENTOS]

* ‘which iptables’ – gives location where the binary is installed
* ‘whereis -b iptables’ – searched only for binary files.
* ‘updatedb’ – building local cache file
* ‘locate table’ – find all binary files on system containing “table”
* ‘find /etc -type f -name ‘passwd’ – find all files with name ‘passwd’
* ‘find / -mtime -3’ – files modified in last 3 days
* ‘find . -name ‘\*.txt’’ -exec rm {} \; - to find files with txt extn and remove them.

***WC, SPLIT, CAT and DIFF Commands***

* ‘cat’ command is used to concatenate two files.
* ‘cat file1 file2’ or ‘cat file\*’
* ‘wc <option> <filename>’ (word count) [-l = number of lines in that file]
* ‘cat file\* | wc -l’ will return line count of all concat.
* ‘split -l 2 file1’ – splits content with specified number of lines and places into separate files.
* ‘diff <filename1> <filename2>’ – will give which lines are different

***Streams(stdin, stdout,stderr) and Redirects***

* A stream is what displays on the screen, we have std input, output, and error.
* Stdin – what ever you are typing on the terminal
* StdOut – what information is display back after successful execution
* StdErr – Error diplay after unsuccessful execution
* Put stderr into a file ‘ls errorfile 2> testfile’
* ‘/dev/null’ is nothing, if we push any thing in to that then it becomes garbage.
* ‘cat xaa xab nofile > mystdopt 2> mystderr’ – simultaneously put data into two locations
* ‘cat xaa xab nofile > mystdopt 2> &1’ – write into the same file
* ‘set -o’ command will list all items which we can turn on and off.
* ‘noclober’ when set on, we cannot overwrite an existing file, .
* ‘set -o noclobber’ turn on , ‘set +o noclobber’ set off.

***Pipes***

* Will take input from one command and put it into the other command.
* ‘ls /etc| sort -f’ [-f = ignore case]
* ‘Less’ is an editor.
* ‘ | ’ PIPE

***grep, egrep, fgrep***

* ‘fgrep’ to speedup searches and ‘egrep’ to use extended regular expression searches.
* ‘grep <search\_string> <filename> ’ or ‘cat testf | grep <string>’
* ‘grep ^hello testf’ [^ = will search for line which start with hello]
* ‘grep -c ^hello testf’ [-c = would give count of the result]
* ‘grep hello$ testf’ [^ = will search for line which end with hello]
* If we need to check for list of characters use ‘[ ]’, ‘grep [hdkf] testf’. We can add ^ or $.
* We can pass ‘-‘ in ‘[]’ , ‘grep [a-g] testf’. We can pass numbers, ‘grep [1-9] testf’
* We can ask grep to read a file, ‘grep -f grepinput testf’
* ‘grep -lr cron /etc’ [-l = gives the only the filenames which include the key in the file rather than the content along, -r= recursiv]
* *Egrep [grep -E]*
  + Use extended regular expressions, can use the different characters
  + ‘egrep ‘hello.\*world’ testf’ – list lines which contains hello and world in it.
  + ‘egrep -i ‘hello.\*world’ testf’ – this will ignore case and list.[-i = ignorecase]
  + ‘egrep ‘hello|world’ testf’ – list lines which contains hello or world in it.
  + ‘egrep -v ‘hello|world’ testf’ – list lines which doesn’t contains hello or world in it.[-v = give not case]
  + ‘egrep ‘hello|world’ testf | grep -v jeff’ – list all lines with hello or world but no jeff in it.
  + ?, {} need to check
* *Fgrep [grep -F]*
  + It will not interpret anything, but just match it literally.,
  + Less cpu is usage, used when resources are concern, speed is concern and we have large data sets to work on.

***CUT***

* ‘cut -f1 -d: <filename>’ [-f1 = which part you want to display, -d: = specifying ‘:’ as delimiter.]

***sed Stream Editor***

* Stream editor for filtering and transforming text, take a pattern or regular expression to search thru a file changing the occurrences or manipulating the pattern in one way or the other.
* ‘sed ‘s/fulltime/parttime/’ team’ – we will search for full time and replace it with parttime in team file and display it on stdout. [s - substitution]
* ‘sed ‘s/fulltime/parttime/w promotions.txt’ team’ – will write the values which were promoted.
* ‘sed ‘/fulltime/w promotions.txt’ team’ – write people who are full time into the text. [w - write]
* If we don’t need stdout, then put it in “/dev/null”
* ‘sed ‘0,/parttime/s/parttime/promotion/’ team ’ – will display output by replacing the partime to promotion at the first occurrence of parttime.
* ‘sed ‘s/<[^>]\*>//’ team’ – [<[^>]\*> = starts with ‘<’ and it doesn’t have ‘>’ beside it, and check any content before ‘>’ ; here ^ = doesn’t denote start of line but not present]

***Tee command***

* Tee helps to reads from stdin and write to stdout and files.
* ‘ls | tee filename’ – this is show results on stdout and will write into the file when compared to ‘ls > filename’ which stores value in filename not showing on stdout.
* We can write data in multiple files – ‘ls | tee filename filename2’
* Rather than replacing the content, if we want to append ‘ls | tee -a filename’ [-a = apnd]
* ‘ls f\*| wc -l|tee count.txt’

***Using Test/Tests on the Linux command line***

* ‘test’ command allows us to check file types and compare values.
* When it is true, it returns ‘0’ and if false then returns ‘1’
* ‘test -f file1’ to check if this is a file or not but will not result displayed for that we need to specify as ‘test -f file1 && echo “true”’ if it true then echo will be executed, else nothing would be displayed. [-w for write permission]
* ‘test 1 -gt 2 && echo “true”’ nothing is displayed as false, ‘test 3 -gt 2 && echo “true”’ will display true.
* If we want to display on false then ‘test 3 -gt 2 && echo “true” || echo “false”’
* We can do it other manner, ‘[5 -eq 5]; echo $?’ will display the result 0 or 1.
* Many more options, explore ‘man test’.

Check for Text on a website from the Linux command line using curl & test

* Curl basically gets the data for the website in html format.
* ‘curl -s google.com | egrep -ci “301 moved” > /dev/null && echo “file moved” || echo “false”’ -----
  + ‘curl -s google.com’ – will website data
  + ‘egrep -ci “301 moved”’ – extended regular expression, [-c = count, -i= ignorecase]
  + Moving data into null and printing some message based on result of egrep.
* If we want to send it to email then
* ‘curl -s google.com | egrep -ci “Internal Server Error” > /dev/null && mail -s “internal server error” [mshah@nisum.com](mailto:mshah@nisum.com)’

# Day 4:

***Writing a For Loop On the Command File***

* “ `(back tick)” = execute command inside a sub shell and paste that contents inside the for loop.
* ‘for name in `ls`; do echo $name; done;’
* ‘for line in `cat file1`; do echo $line; done;’
* ‘for line in `cat file1`; do echo $line+helloworld; done;’
* ‘for line in `cat file1`; do touch $line; done;’

Exercise