============================================================================================

**Week: 1**

1. **cd**  -> change directory
2. **pwd**  -> present working directory
3. **ls**  -> list the files in the current directory

**ls -l**  -> list the files but in long listing (detailed manner)

**ls -a**  -> list the files including hidden files.

**ls -la**  -> list the files including hidden file but in long listing.

1. **cal**  -> shows the calender (current month).

**cal 2 2023** -> shows the calender but of specific month and year.

**cal 2023** -> shows the whole calender of 2023 year.

**cal 1**  ->

#Bonus calender in matrix range from from 1 - 9999

1. **date**  -> current date and time (respect to server)
2. **who** -> people connected to the same node.
3. **w**  -> detailed view of the who command including the server up-time.
4. **whoami** -> current username
5. **who am i**  -> same thing as who but just for us.
6. **clear**  -> clears the screen.
7. **exit**  -> logout from matrix.

**logout** -> same thing

============================================================================================

**Week: 2**

1. **tree** -> show the tree view of current present working directory.

**tree -C** -> it shows the tree but in color.

1. **mkdir** -> make directory.

**mkdir** **-p** -> create parent and sub/child directories both.

1. **rmdir** -> remove directory (empty).
2. **rm** -> remove file.

**rm -r** -> remove directory (including non-empty) and file.

1. **cp** -> copy files.

**cp -r**  -> copy directories (including empty and non-empty) and files.

1. **mv** -> move files or directories (both empty and non-empty).

**# -i** Prompt for confirmation if the destination path exists. can be used with cp mv rm.

1. **touch**  -> create text file.
2. **vi** -> text editor.

**ESC**  -> command mode.

**i**  -> insert mode.

**:w** -> Save.

**:x**  -> Save and exit.

**:q**  -> exit if no changes made.

**:q!**  -> exit & discard any changes.

1. **nano**  -> text editor.
2. **cat** -> text reader.

**cat -n** -> shows the content of file with line number.

**more**

**less** -> similar to more but also allows forward and backward movement.

**head**

**head -5** -> shows the first five lines.

**tail**

**tail -5** -> show the last five lines .

1. **file** -> tell the type of file. (Helpful for the files which don’t have any e
2. **find**  -> search for the files in a directory hierarchy.

Ex: find ~ -name “output.txt”

1. **diff**  -> compare files line by line.

**diff f1 f2 -y** -> compare two files side by side.

1. **sort** -> sort lines of text file.
2. **uniq**  -> omits same lines.
3. **grep**  -> matches the pattern.

========================================================================

**Week: 3**

1. **Absoulute path ->** /home/pgarg13/slg/output.txt
2. **Relative path** -> output.txt
3. **Relative to home path** -> ~/slg/output.txt
4. **File name expansions**
   1. **\*** -> 0 or more characters
   2. **?** -> exactly one character(any)
   3. **[ ]** -> one character from the range
   4. **[! ]** -> except those characters in the range
5. **echo** -> display a line of text.

**echo red** -> display text red.

**echo red\*** -> show the files starting with red.

**echo $USER** -> show current username.

**echo '\* $USER \*'** -> display \* $USER \* //single quotes strong quotes

**echo “\* $USER \*”** -> display \* pgarg13 \* // double quotes weak quotes

1. **fc** -l -> list previous 16 commands
2. **history** -> list all the commands
3. **who | wc -l** -> gives the number of users or we can say lines in who command.

**Creating a variable:**

1. **number=5** -> variable name is number, and its value is 5(no spaces before or after = sign)

**#** The variables thatyou create exist for the lifetime of terminal if you close it is gone

**school=”Seneca”**

**myseneca=“My school is $school”** -> it will print: My school is Seneca

**echo $myseneca**

**myseneca=’My school is $school’** -> it will print: My school is $school

**37. echo $number**  -> prints the value of number on the screen.

**To store commands in a variable:**

1. **longListing=$(ls -l)**

**longListing=`ls -l`** -> both are same thing

**39. echo $longListing** -> it will execute the ls -l command, but output will be messy

**echo `**

**# To prevent that**

**echo “$longListing”** -> put it in double inverted commas for organised output.

**To print something and execute a command inside it:**

**40. echo “Today’s date is `date`”** -> it will print out : Today’s date is Thu Feb 9 15:04:54 EST 2023

===========================================================================

**Week 4:**

**File Permissions:**

**41. chmod -> change mod to change the file permisson**

**symbolic method:**

**42. chmod -r file.txt** -> removes the read command from user, group and others for file.txt

**43. chmod -x file.txt** -> removes the execute command from user, group and others for file.txt

**44. chmod -w file.txt** -> removes write command from only user

===========================================================================

**Week 5**

**45. grep**  -> print lines that matches the pattern

ex: **grep ‘Mark’ file.txt** -> shows the line which has mark in it.

**grep -i**  -> show the pattern ignoring the case.

**grep -v**  -> reverse searchs (or inverted search)

**grep -c** -> prints the number of lines that matches the pattern.

**grep -w** -> prints the line that matches the pattern exactly nothing less nothing more.

**grep -n** -> to get the line number which has the pattern matched.

**grep** -r -> recursive search

**ex: grep -r ‘ford’ .**  -> it will look for word ‘ford’ in all the files and print the lines (even nested files)

**46. sort**  -> sort lines of text file(by default first column and ascending order)

**sort -k3**  -> sort the lines of text file based on third column.(but this will not run as if we mention column then also mention delimiter)

# we have to mention by which delimiter column is seperated so,

**sort -t’,’ -k3** -> sort the file based on the third column.

**sort -n**  `-> sort numerically

**sort -u** -> sort but get rid of duplications.

**sort -R** -> sort in random order.

**sort -f**  -> sort file but ignore case.

**52. tr**  -> translate or delete character

# but it only works if we use < symbol before the file name but with pipline it is ok

For ex:

tr ‘,’ ‘-‘ < data.txt -> it replaces all , with –

**tr -s ‘ ‘ ‘,’ < cars.txt** -> it replaces all space with commas and also remove the duplicate of ,

**53. cut** -> cut the output from the file (doesn’t change the original file)

**For ex**:

**cut -d’,’ -f1,2 data.txt** -> print the first and second field from data.txt which is delimited by ‘,’ .

**54. wc**  -> gives the count for the file.

**wc -m** -> gives the char count for the file.

**wc -l** -> gives the line count for the file.

**wc -c** -> prints the byte count for the file.

**55. >** -> redirect from left to right.

**56. <**  -> redirect from right to left.

**57. 2>** -> redirect error to file(this will overwrite the file)

**58. 2>>** -> redirect error to the file but at the end

# EX:

wc nofile file1 > wordcount 2>> errorlog (combination of redirection of standard output and error )

**59. >&** -> redirect both error and output in same file

**60. |**  -> redirect standard output.

**61. tee** -> split the flow to file and standard output,

**tee -a**  -> when splitting to file append to the end of

=====================================================================

**Week 6**

**62. mail *youremail*** -> to send email( if you hit enter after writing a line you cannot go back to previous line to edit) ctrl+d to send the mail

**mail -s “subject of your email” *youremail***

**mail -s “subject of your email” *youremail* < file.txt**

**mail -s “subject of your email” -a attachment.txt youremail < file.txt**

**63. scp source destination**

For ex:

scp file.txt [username@myseneca.ca:~/zaa/transfer](mailto:username@myseneca.ca:~/zaa/transfer)

**scp -r source destination**

64. **sftp yourmatrixserver ->** opens your matrix server on you cmd ( just for file transfer and basic stuff not pico or anything )

**commands to run when you open matrix on you local cmd.**

**local pc matrix or server**

**lls ls**

**lpwd pwd**

**lmkdir mkdir**

**lcd cd**

**put file.txt** -> upload file from pc to server .

**get file.txt** -> upload file from server to pc.

**exit** -> to exit from sftp.

===========================================================================

**Week 8**

**Linking Files**

**Hard Links (ln) – only works for files. Have same inodes. both file have same permissions and everything**

**65. ln source destinaton**

**touch F1**

**ln F1 F2**  -> link F1 and F2

**ls -li** -> to check which files are linked (i means inode) it will also show inode the files with same inode are linked.

( in hard link it doesnot



matter which file I delete.



the data would still be there.)



**Symbolic Links (ln -s) - works for both dir and files. have different inodes. The file (F2) that I linked with orginal file (F1) has all permissions (lrwxrwxrwx)**

**66. ls -s source destination**



**ls -s F1 F2**

**(If I delete F1 data of F1 is gone,**

**F2 is not gone but**

**it will point to F1 path.**

**If I recreate F1 then it is going**

**to look again for F1 and it**

**will point to the newly created)**

**# path matters when creating link**

**Processes:**

**67. ps** -> show the processes running (PID process ID each process has unique)

**ps -x** -> shows all processes on same node of all instances of the terminal.

**ps afux** -> show all the process for all the user on same node.

**ps -U userID** -> shows the process of specific user.

**68.** **kill PID**  -> to stop or kill the process .

**kill -9 PID** -> force stop or kill.

**69.** **tail -f log.txt** -> it will not exit the tail command it will update the tail output in real time.

**70. jobs** ->shows the process running in background.

**jobs -l** -> show the process with PID.

**71. fg #ofjob**  -> bring the specific process in foreground

**72.** **command &** -> send the command to background

**73. top**  -> shows the memory usage of all process in real time

**74. du -h** -> Disk usage

**75. df -h ->** Disk FIle System

===========================================================================

**Week 9**

**Regex**

**special Characters**

. -> single character

\* -> zero or more occurence

.\* -> any number of characters

^ -> begin with

$ -> end with

[] -> represent single character.

[^] -> exclude that character.

===========================================================================

**Week 10**

**76. sed** -> manipulate text in the file

**sed ‘1,4 p’ file.txt** -> print whole file and print line 1 to 4 twice ( p stands for print)

**sed -n ‘1-4 p’ file.txt** -> print just line 1 to 4 (mostly with p we use -n option)

#Note: order of 1 and 4 matters 1-4 is correct but 4-1 is incorrect

**sed ‘1-5 d’ file.txt** -> delete line 1 to 5 (means show rest of the lines).

**sed ‘5 q’ file.txt** -> print lines from beginning to line 5( or we can say quit printing at the line provided)

# Note: no ranges in quit so ‘1,5 q’ is wrong

**sed -n ‘/boy/ p’ file.txt** -> print lines with word boy in it( remember -n option)

**sed -n ‘/boy/I p’ file.txt** -> print lines with word boy in it but ignore case(i is capital)

**sed ‘s/a/X/’ file.txt**  -> replace the first a in each line with X.

**sed ‘s/a/X/g’ file.txt**  -> replace all a’s with X

77. awk ‘{print $1, $2}’ cars.txt -> print first and second field (comma is imp it will iclude space between fields)

**$NF** -> last column

**NF** -> number of columns in each row

**$0** -> all columns

**NR** -> row number

**awk ‘NR==1’ cars.txt** -> print row no. 1

**awk ‘NR <=5’ cars.txt**  -> print from beginning to line 5

**awk ‘NR == 5 || NR == 7’ cars.txt** -> print line no. 5 and 7

**awk ‘$5 <= 3000’ cars.txt** -> print the row which has 5th field less than equal to 3000

**awk ‘/ford/’ cars.txt** -> print the lines with ford in it

**awk ‘/chevy/ {print $0}’ cars.txt** -> print the line which has chevy in it

**awk ‘/chevy/ {print $1, $2}’ cars.txt** -> print line with chevy but only show 1st and 2nd column

**awk ‘/5$/ { print $1, $2}’ cars.txt** -> print column 1 and 2 of line which ends with 5.

**awk -F, ‘{print $1, $4}’ cars.txt** -> print column 1 and 4 and the file is , delimited.

**===========================================================================**

**Week 11**

**Scripting ->**

**#!/bin/bash**

**# This is a comment**

**echo “hello world”**

change the permissions of the file. chmod +x

just write the file name it will run it.

**echo -n “My first hello world program”** -> this will read in the same line as echo not in new line

read variable

**echo “the first argmnet is $1”** -> replace the $1 with the first argument provided while running the script

**echo “the tenth argument is $10”** -> this is wrong it will not print 10th argument.

**echo “the tenth argument is ${10}”**  -> use currly braces when writing two digit arguments

**echo “ the total number of arguments are $#”** -> display total number or arguments.

**if [condition]**

**then**

**/Do something/** -> if statements

**fi**

**if [condition]**

**then**

**/Do something/** -> if else conditon

**else**

**/Do something/**

**fi**

**$PWD** -> present working directory

**$HOME** -> home directory

**$USER** -> your username **env** -> to view environment variables

**echo $?** -> shows the exit code of script ran previously

**0**  ->this exit code means successful execution

**-lt** -> less than

**-le** -> less than or equal to

**-eq** -> equal to

**-gt** -> greater than

**-ge** -> greater than equal to

**for VAR in \_ \_ \_**

**do**

**/do something/**

**done**

**$@ -> stores every argument provided by user when running the script**

**$\* -> same as previous**

**“$\*” -> group all the arguments and treat it as one argument**

**$# -> number of arguments**

**while [conditon]**

**do**

**/do something/**

**done**

**$(( $variable +1 )) -> to do some calculations**

**Example:**

#!/bin/bash

low=$(echo $1 | cut -d- -f1)

high=$(echo $1 | cut -d- -f2)

range=$((high -low +1 ))

secretNumber=$(( low+RANDOM%range))

count=1

while true

do

echo “Enter a number between between $low and $high: ”

read guess

if [ $guess = $secretNumber]

then

echo “----------------CORRECT-----------------"

echo “You guessed it in $count tries”

echo “----------------------------------------------"

exit 0

elif [$guess -lt $secretNumber]

then

echo “Too low! Try guessing a higher number”

else

echo “Try guessing a lower number”

fi

count=$(($count +1 ))