***Unix Practicals***

**Experiment No. 1 : *Ubuntu Installation*** ☕

**Experiment No. 2 : *Basic Commands***

1. **echo-** Echo is a Unix/Linux command tool used for displaying lines of text or string which are passed as arguments on the command line.



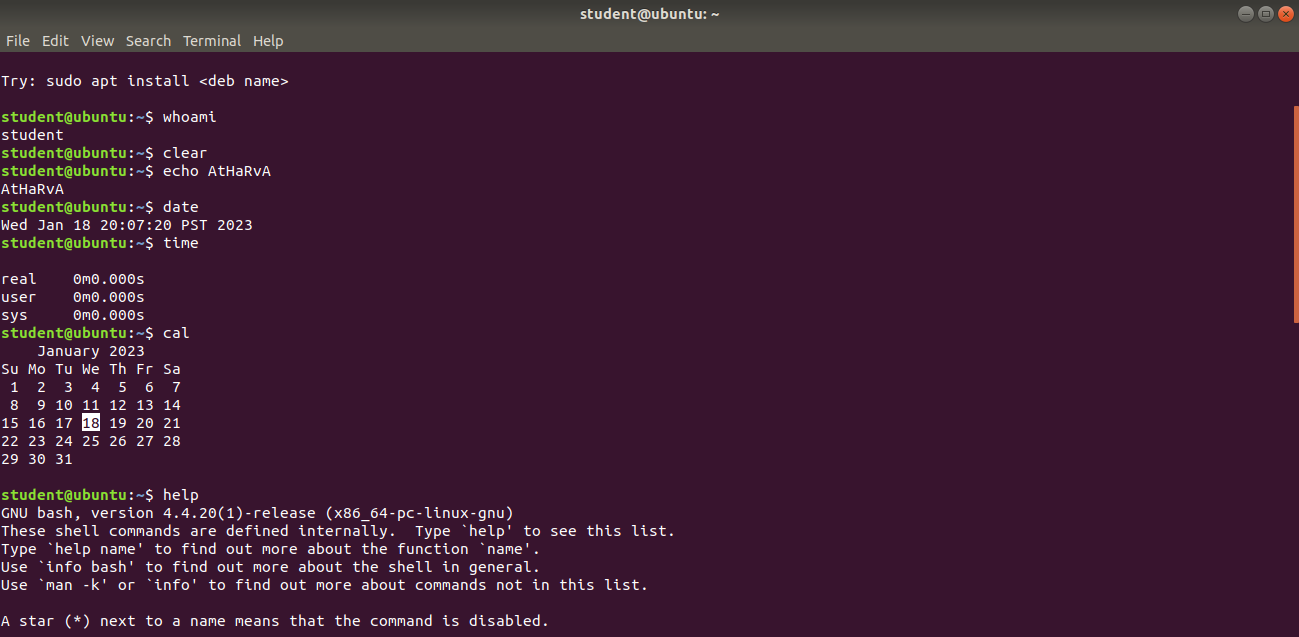
1. **clear -** This command is used to clear the contents of the terminal screen.



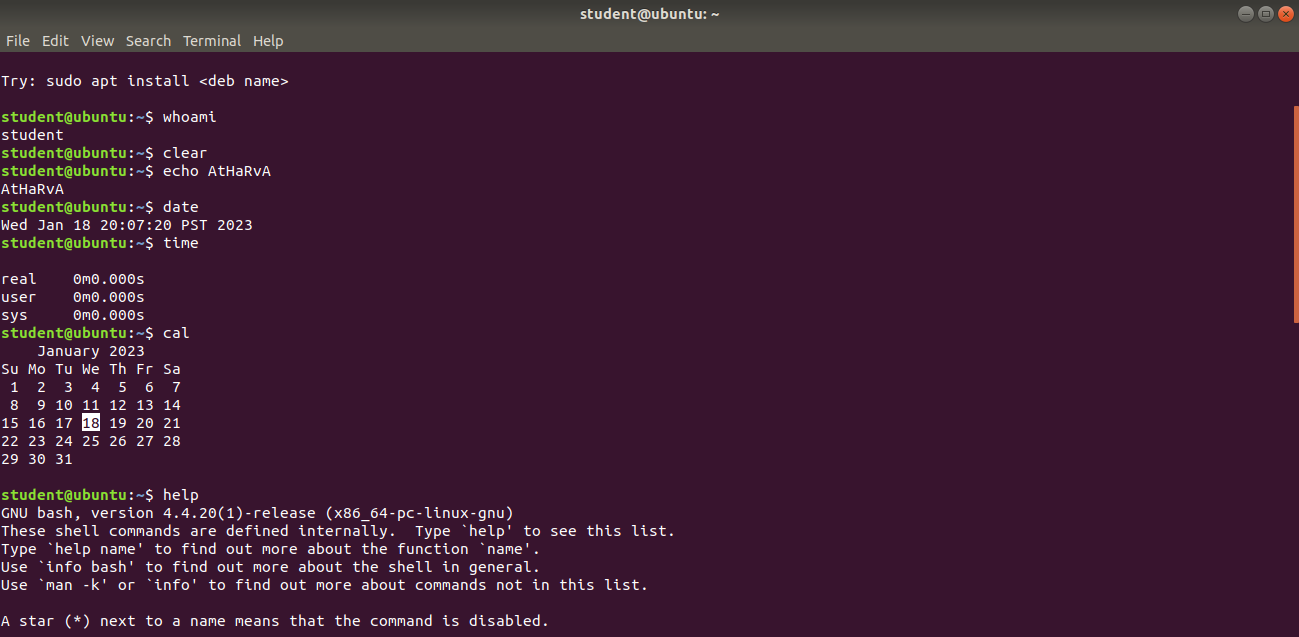
1. **exit -** This command is used to exit from the current shell or terminal.



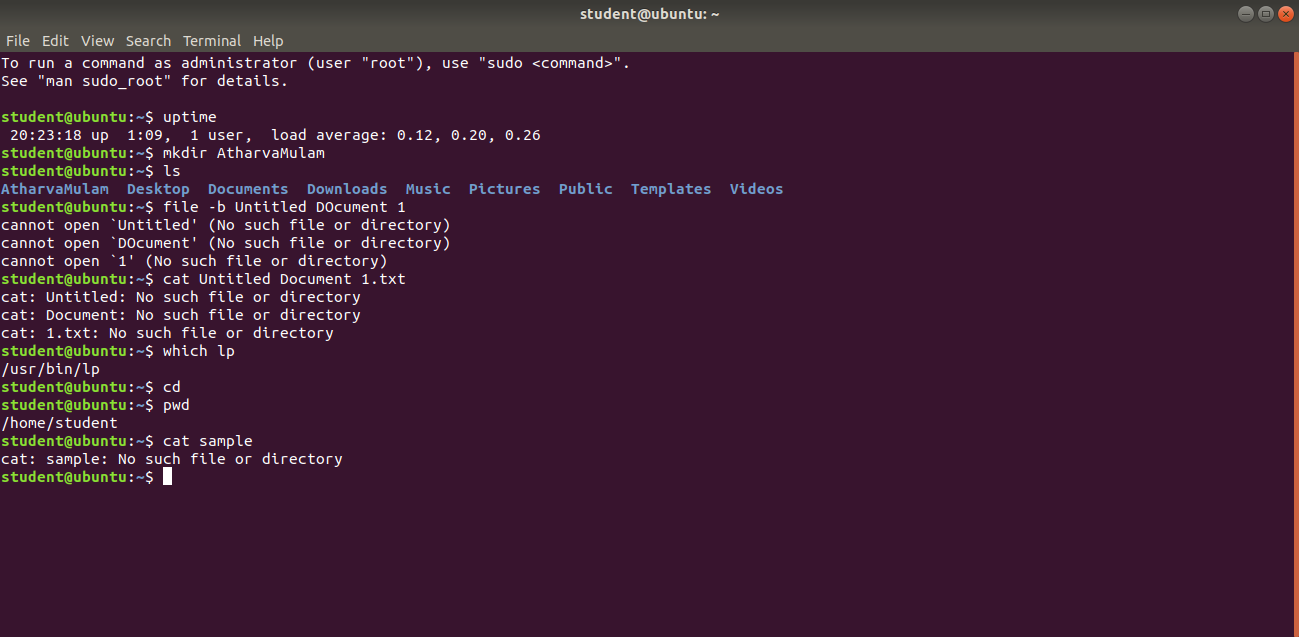
1. **time -** This command is used to measure the time it takes to execute a command or a script.

****

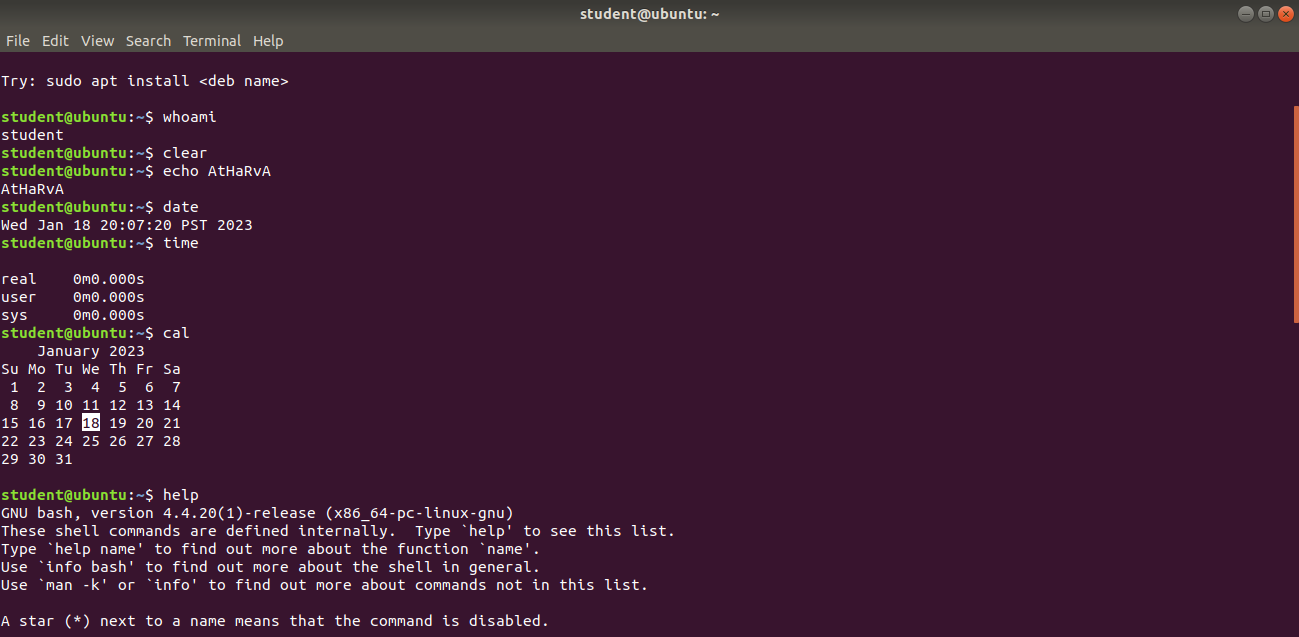
1. **date-** This command is used to display the current date and time.

****

1. **uptime -** This command is used to display how long the system has been running and the system load average.

****

1. **cal -** This command is used to display the calendar for the current month or a specific month and year.

****

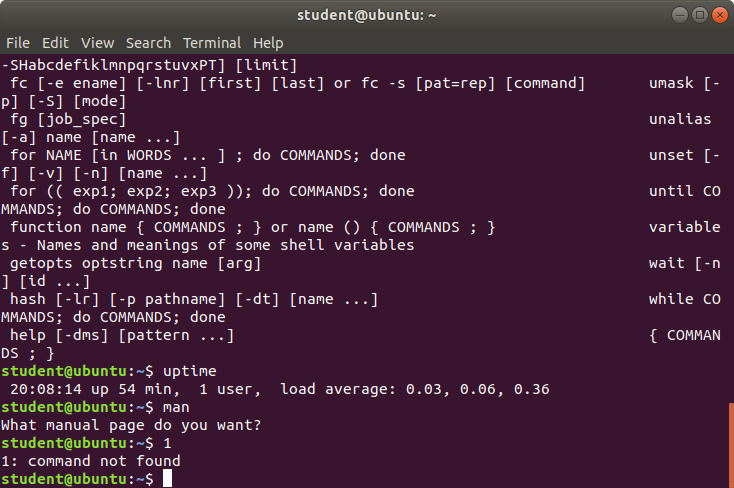
1. **cat -** This command is used to display the contents of a file on the terminal.



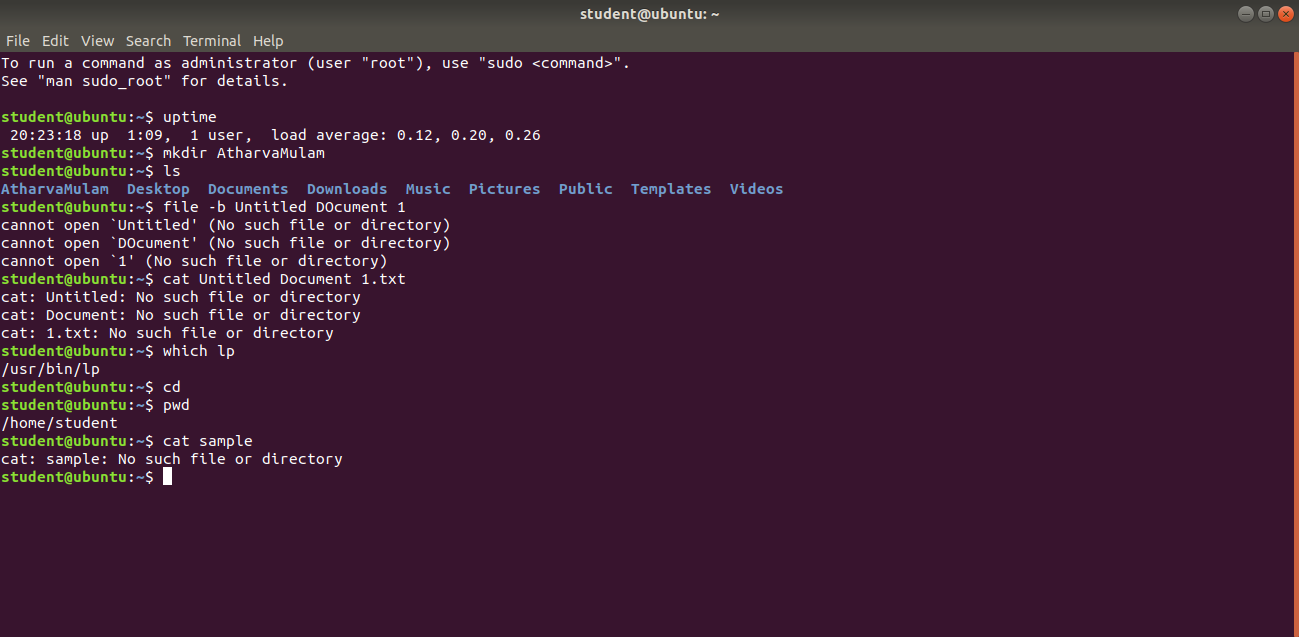
1. **tty -** This command is used to display the terminal device name.



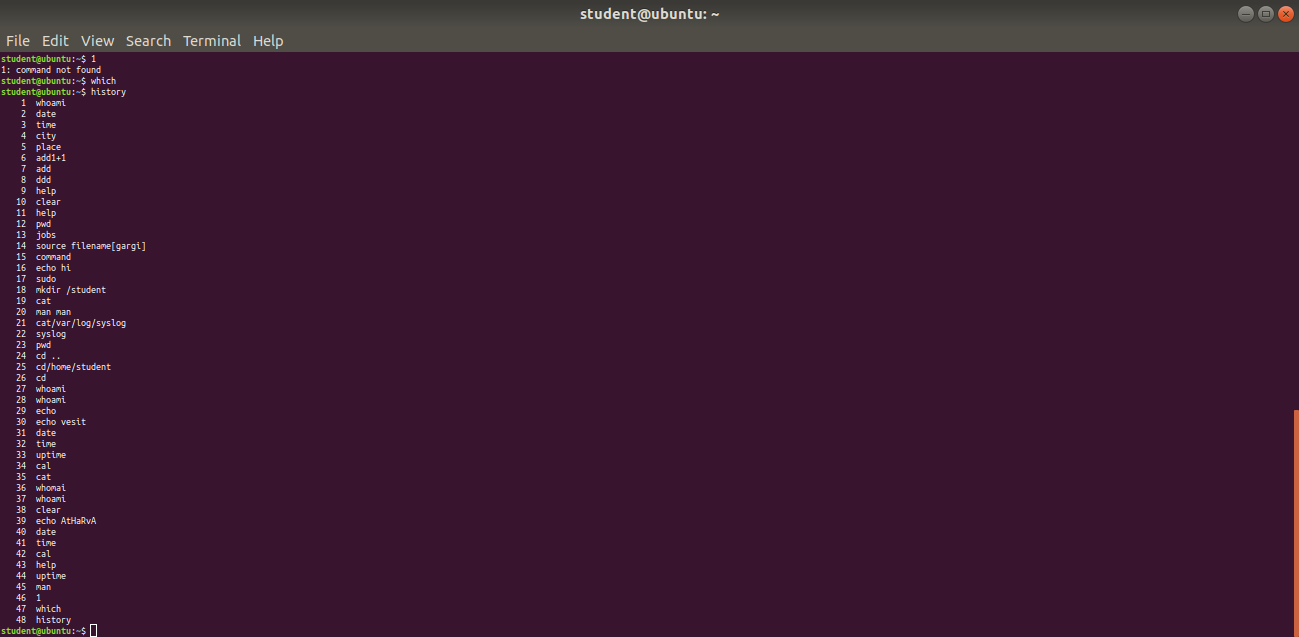
**10)** **man -** This command is used to display the manual pages of a command or a program.

****

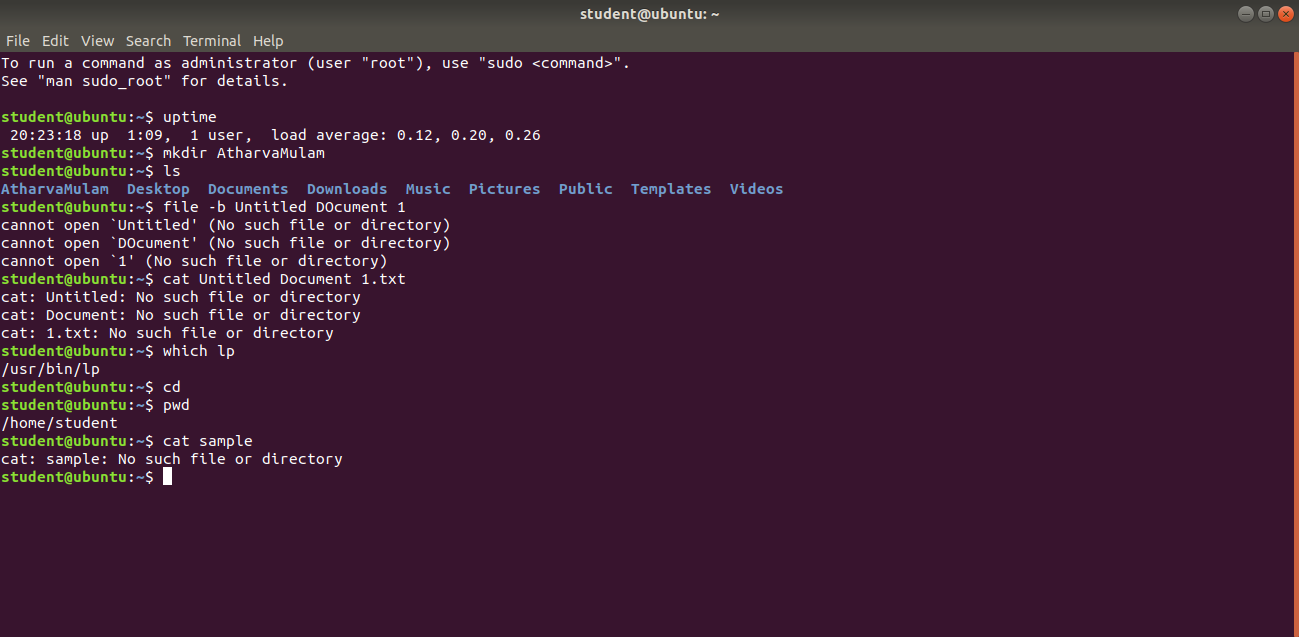
**11)** **which -** This command is used to display the location of a command or a program.

****

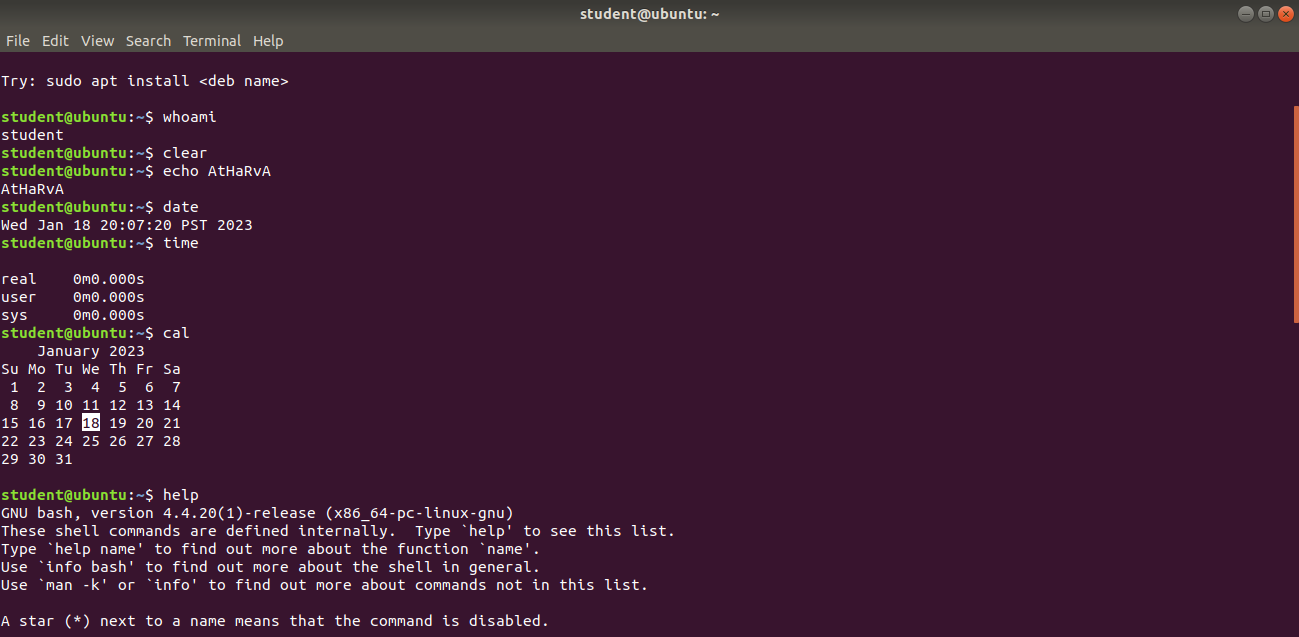
**12)** **history -** This command is used to display the previously executed commands.

****

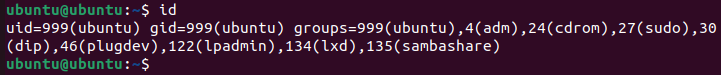
**13)** **pwd -** This command is used to display the current working directory.

****

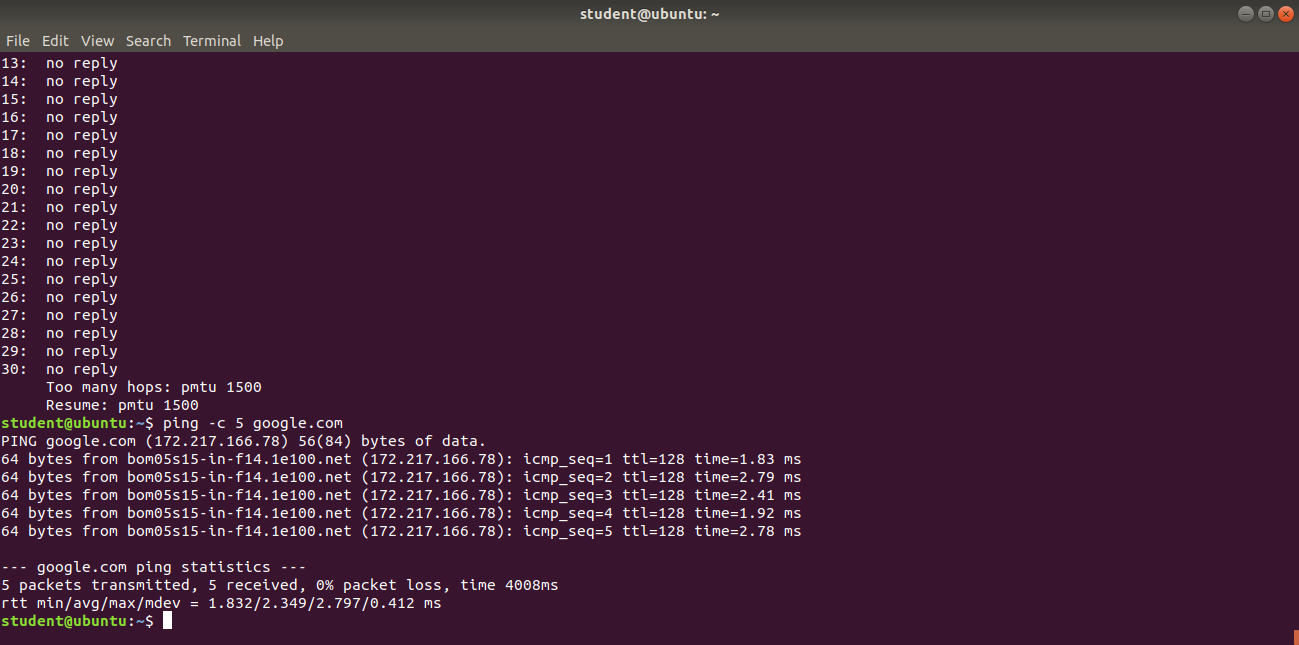
**14)** **whoami -** This command is used to display the username of the current user.

****

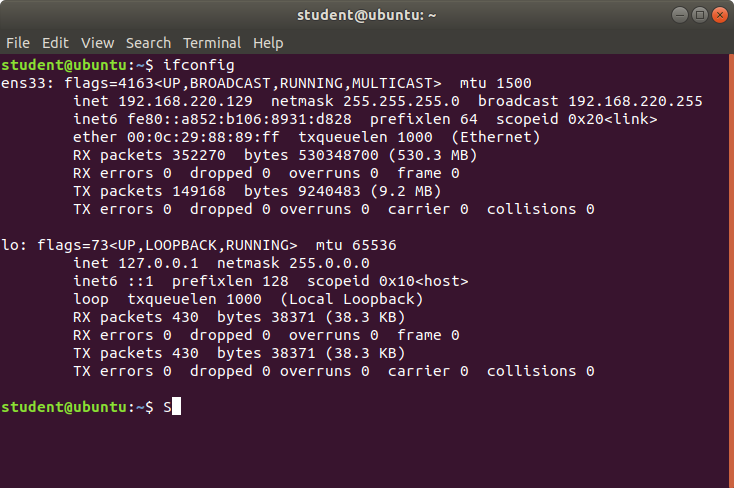
**15)** **id -** This command is used to display the user and group IDs.



**16) ping -** This command is used to check the connectivity to a network host or IP address.



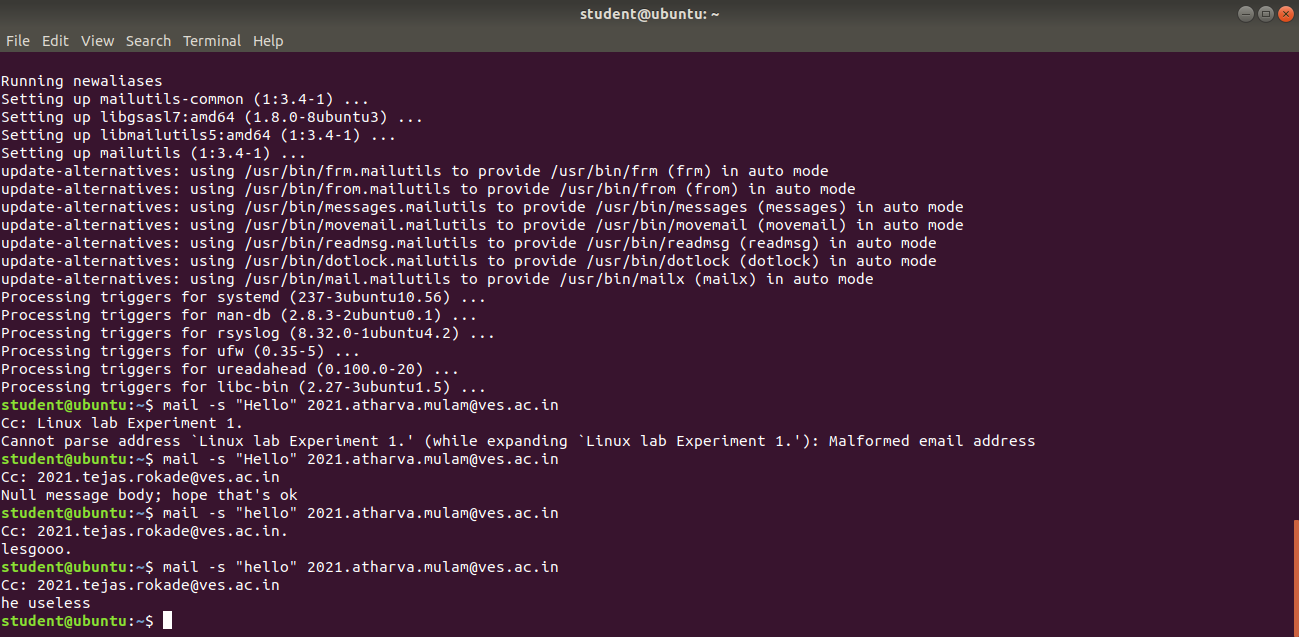
**17) ifconfig -** This command is used to check the connectivity to a network host or IP address.

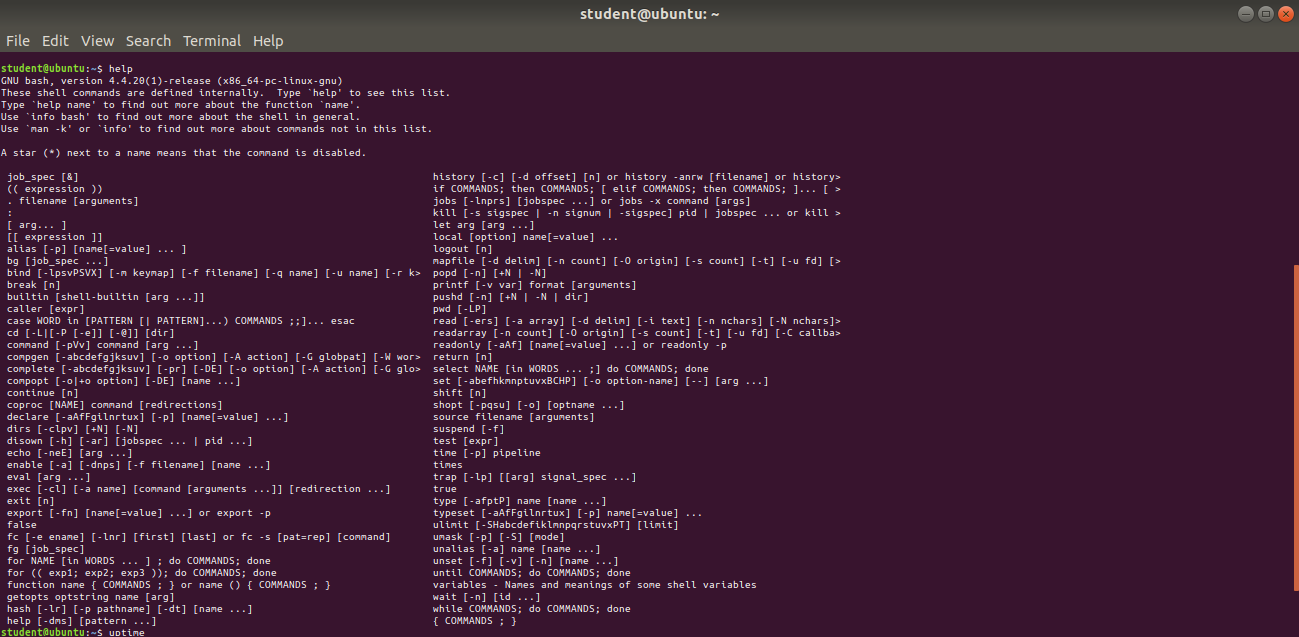


**18) pr -** This command is used to format files for printing.



**19)** **mail -** This command is used to send and receive email messages from the terminal.

****

**20)** help****

**Working with VI editor** [2021.gaurang.raorane@ves.ac.in](mailto:2021.gaurang.raorane@ves.ac.in)

**Experiment No. 3 : *Commands for File System Management and User Management***

1. ***Study of Unix file system (tree structure), file and directory permissions, single and multiuser environment.***

**1.** Unix file system is a logical method of organizing and storing large amounts of information in a way that makes it easy to manage. A file is a smallest unit in which the information is stored. The Unix file system has several important features. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system. Files in the Unix System are organized into a multi-level hierarchy structure known as a directory tree. At the very top of the file system is a directory called “root” which is represented by a “/”. All other files are “descendants” of root.

**2.** Every file and directory in your UNIX/Linux system has the following 3 permissions defined for all the 3 owners discussed above.

**Read:** This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to list its content.

**Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.

**Execute:** In Windows, an executable program usually has an extension “.exe” and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it

**3.** In a single-user environment, only one user has access to the computer and its file system. In this scenario, file and directory permissions are not as important, since there is only one user who needs access to the files.

In a multi-user environment, multiple users have access to the computer and its file system. In this scenario, file and directory permissions play a crucial role in determining who can access and modify files. This helps to ensure the security and privacy of sensitive files, and prevents unauthorized access and modification of files by other users.

1. ***Execution of File System Management Commands like ls, cd, pwd, cat, mkdir, rmdir, rm, cp, mv, chmod,wc, piping and redirection, grep, tr, echo, sort, head,tail, diff, comm, less, more, file, type, split, cmp,tar, find, vim, gzip, bzip2, unzip, locate, etc.***

**1) PWD-**The pwd command writes to standard output the full path name of your current directory (from the root directory).



**2) ls-**The ls command is used in listing contents inside a directory and is one of the few commands beginners learn from the onset.



**3) cat-** It can be used to display the content of a file, copy content from one file to another, concatenate the contents of multiple files, display the line number, display $ at the end of the line, etc.



**4) mkdir-**The mkdir stands for 'make directory'. With the help of mkdir command, you can create a new directory wherever you want in your system.



**5) rmdir-**The rmdir command removes the directory, specified by the Directory parameter, from the system.



**6) rm-**The rm command removes the entries for a specified file, group of files, or certain select files from a list within a directory.



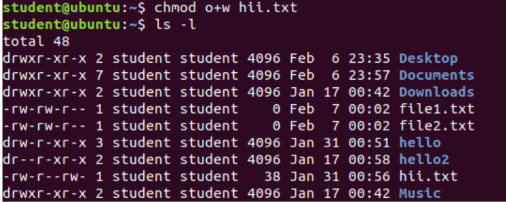
**7) cp-** cp stands for copy. This command is used to copy files or groups of files or directory. It creates an exact image of a file on a disk with a different file name. cp command requires at least two filenames in its arguments.



**8) mv-** mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX.



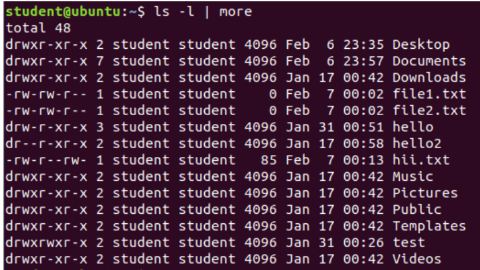
**9) chmod-** The chmod command is used to change the access mode of a file.



**10) wc-** The Linux wc command calculates a file's word, line, character, or byte count.



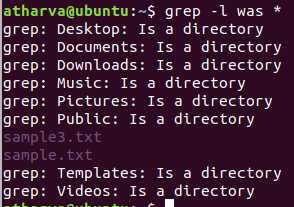
**11) piping-** Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on.



**12) grep-** The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

****

-c : This prints only a count of the lines that match a pattern

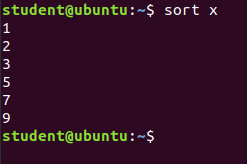


-l Displays list of filenames only.

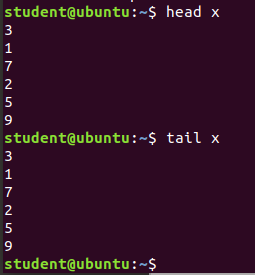
**13) tr-** The tr command is a Linux command-line utility that translates or deletes characters from standard input ( stdin ) and writes the result to standard output ( stdout ).



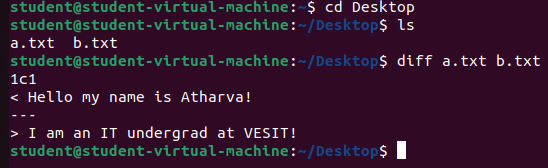
**14) Sort-** sorts the lines of a file.



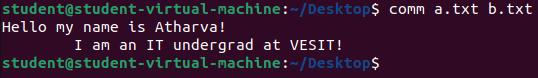
**15) Head and Tail-** Head command is a command-line utility, which prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name. Tail is a command which prints the last few number of lines of a certain file, then terminates.



**16) diff-** diff stands for difference. This command is used to display the differences in the files by comparing the files line by line.



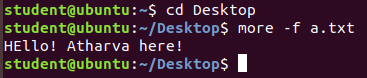
**17) comm -** comm compare two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.



**18) less -** Less command is a Linux utility that can be used to read the contents of a text file one page(one screen) at a time.



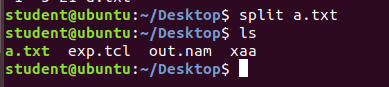
**19) more-** more command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large.



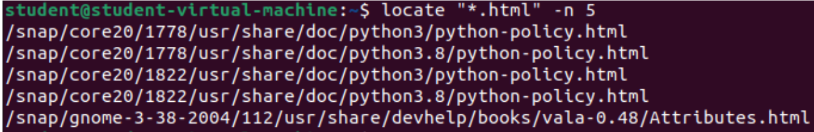
**20) file-**  file command is used to determine the type of a file. .file type may be of human-readable(e.g. ‘ASCII text’) or MIME type(e.g. ‘text/plain; charset=us-ascii’). This command tests each argument in an attempt to categorize it.



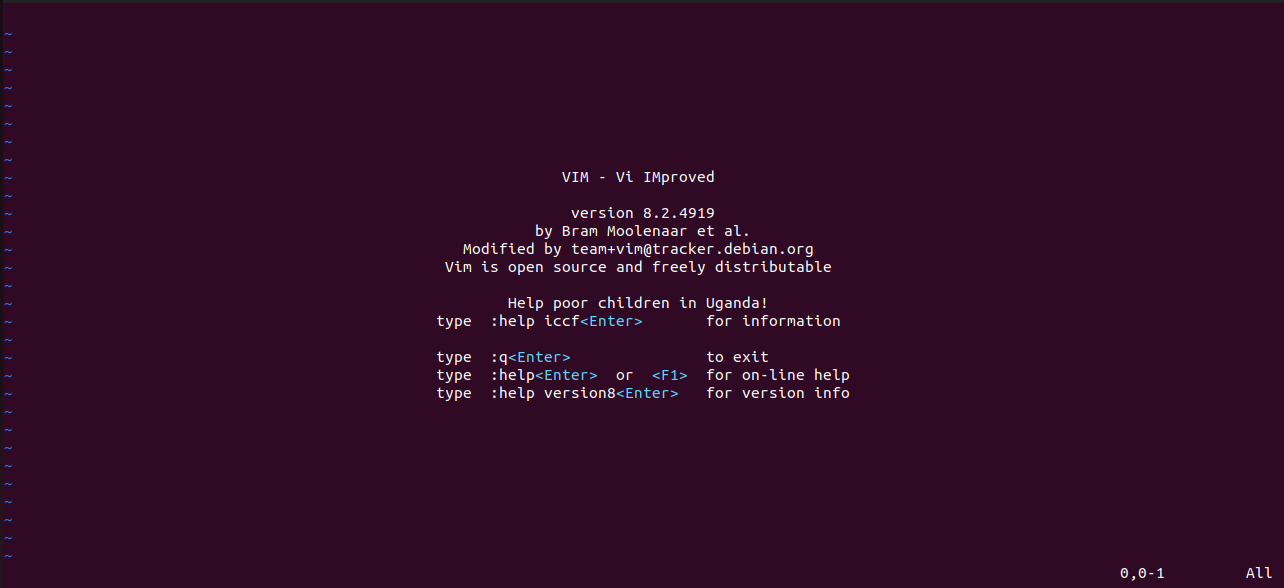
**21) split-** Split command in Linux is used to split large files into smaller files. It splits the files into 1000 lines per file(by default) and even allows users to change the number of lines as per requirement.



**22) locate-** locate command in Linux is used to find the files by name.



**23) vim-** Vim is an advanced and highly configurable text editor built to enable efficient text editing. It supports most file types and vim editor is also known as a programmer’s editor. We can use its plugin based on our needs



***C) Execution of User Management Commands like who ,whomai ,su, sudo,login, logout,exit,passwd,useradd/adduser,usermod,userdel,groupadd,groupmod,groupdel,gpasswd,chown,chage,chgrp,chfn,etc.***

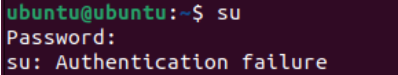
1. **who-** displays information about the users who are currently logged in.

****

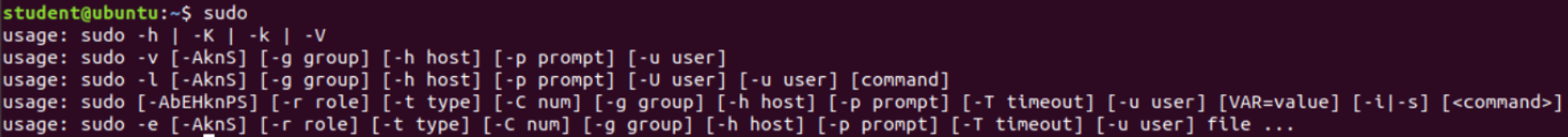
1. **whoami-** displays the username of the current user.

****

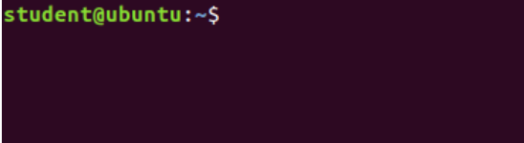
1. **su-** switches the current user to another user, typically the superuser.

****

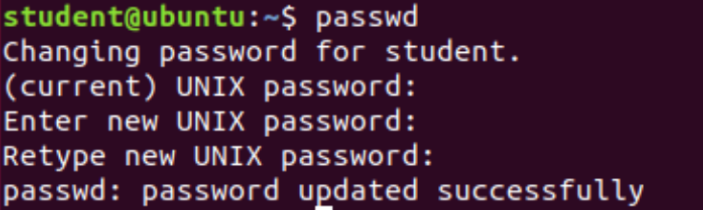
1. **sudo-** allows a user to run a command with superuser privileges.

****

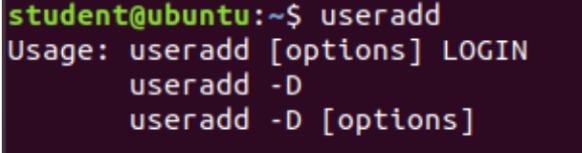
1. **exit-** exit command in linux is used to exit the shell where it is currently running.

****

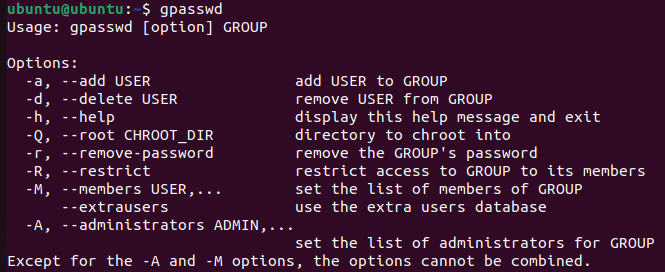
1. **passwd-** The passwd command changes passwords for user accounts. A normal user may only change the password for their own account, while the superuser may change the password for any account.

****

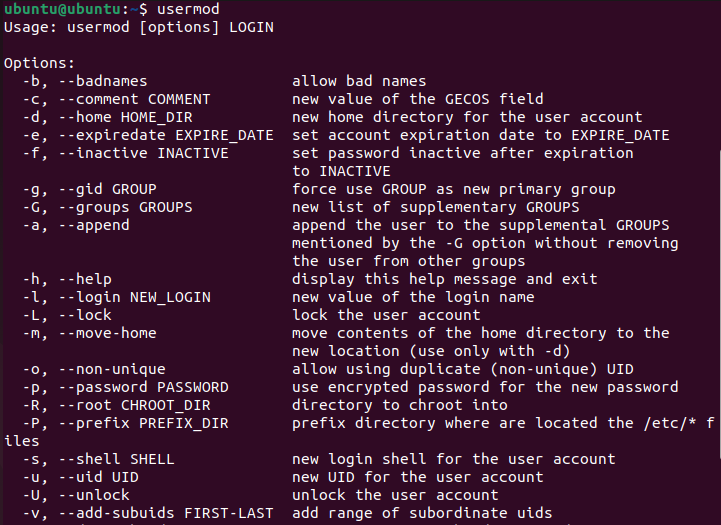
1. **useradd-** useradd is a command in Linux that is used to add user accounts to your system.

****

1. **gpassswd-** gpasswd command is used to administer the /etc/group and /etc/gshadow. As every group in Linux has administrators, members, and a password.

****

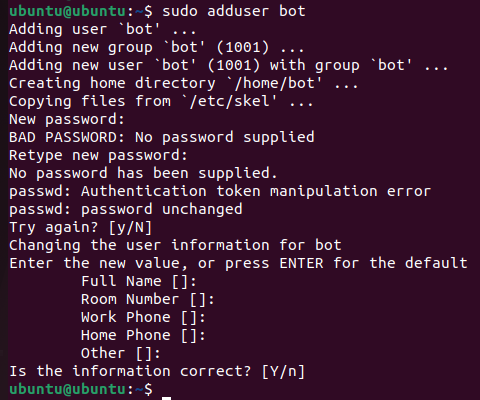
1. **Usermod-** Usermod command or modify user is a command in Linux that is used to change the properties of a user in Linux through the command line. After creating a user we have to sometimes change their attributes like password or login directory etc. so in order to do that we use the Usermod command.



1. **Userdel-** userdel command in Linux system is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.



1. **Useradd-** Useradd is a command in Linux that is used to add user accounts to your system. It is just a symbolic link to adduser command in Linux and the difference between both of them is that useradd is a native binary compiled with system whereas adduser is a Perl script which uses useradd binary in the background.



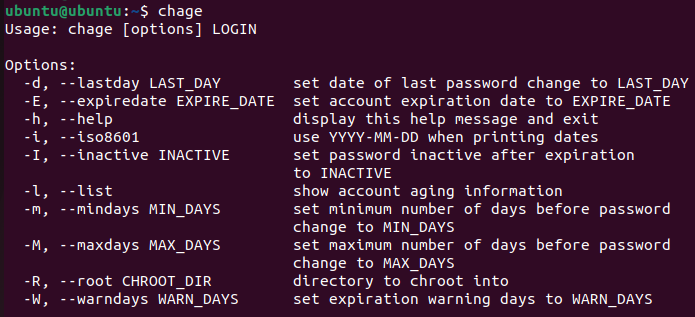
1. **Groupadd-** So using groups, we can group together a number of users, and set privileges and permissions for the entire group. *groupadd* command is used to create a new user group.

****

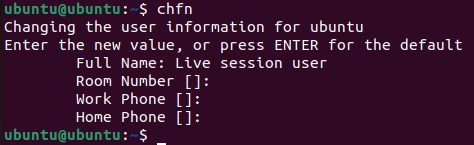
1. **Groupdel-** *groupdel* command is used to delete a existing group. It will delete all entry that refers to the group, modifies the system account files, and it is handled by superuser or root user.

****

1. **chage-** The chage command is used to view and change the user password expiry information. This command is used when the login is to be provided for a user for a limited amount of time or when it is necessary to change the login password from time to time.

****

1. **chfn-** *chfn* command in Linux allows you to change a user’s name and other details easily. *chfn* stands for Change finger. Basically, it is used to modify your finger information on Linux system.

****

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

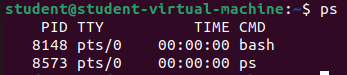
**Experiment No. 4 :*****Commands for Process Management and Memory Management***

1. ***Process Management***
2. **ps-**  ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it’s referred to as a virtual file system.

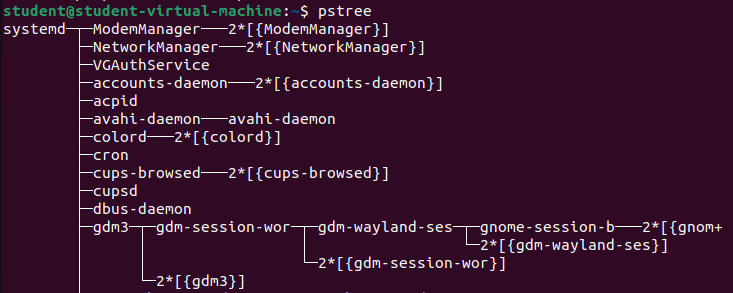








1. **pstree**- **Pstree** command in Linux that shows the running processes as a tree which is a more convenient way to display the processes hierarchy and makes the output more visually appealing.



Aur types:-

pstree -a (include command line arguments in o/p)

pstree -p (to display the PIDs)

pstree -c (to force pstree to expand identical subtrees in output)

pstree -n (to sort processes)

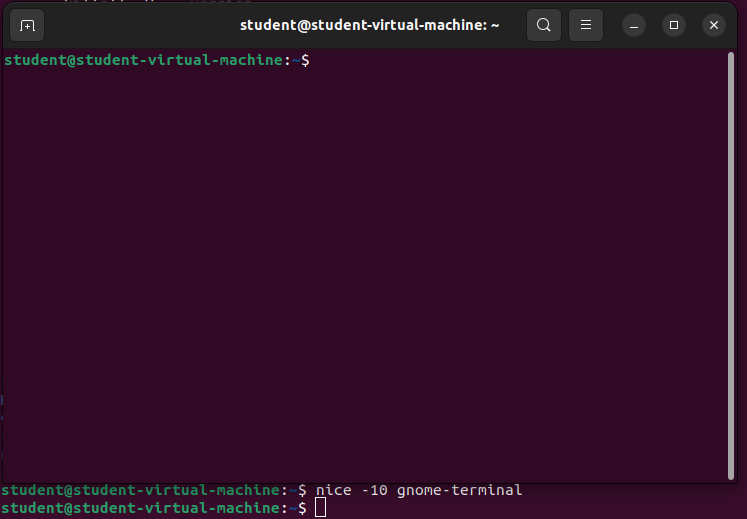
pstree -u (to see who is the user of the program)

1. **nice-** **nice command** in Linux helps in execution of a program/process with modified scheduling priority. It launches a process with a user-defined scheduling priority. In this, if we give a process a higher priority, then Kernel will allocate more CPU time to that process.



The "nice" command in Linux adjusts the scheduling priority of a process, and running it without arguments returns the current niceness value. Niceness ranges from -20 to +19, with a higher value indicating a lower scheduling priority. A value of 0 means the process has the default niceness, running at standard priority, and is not given any special or deprioritized treatment.

**nice -10 gnome-terminal** is used to set the priority of a process.



1. **kill-** *kill* command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. *kill* command sends a signal to a process which terminates the process.



PID 7184 belongs to Mozilla Firefox and it was killed after running kill command.

1. **pkill-** The pkill command uses name of the process instead of PID number. Signal can be send to a process either by typing full name or partial name.



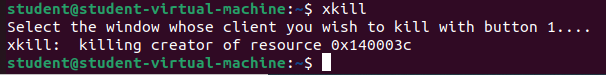
Firefox was killed!

1. **Killall-** This command provides a convenient means of canceling all processes created by the shell that you control. When started by a root user, the killall command cancels all cancellable processes except those processes that started it.

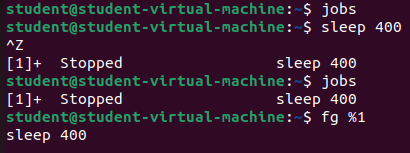
**syntax**- sudo killall processname, it kills all the processes that have the same name.



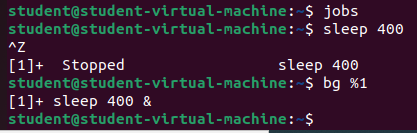
1. **Xkill-** xkill is a command-line utility that can kill the undesired windows on the user's screen. Basically, xkill force the X server to close the connection to the client. This utility kills the programs without providing PID with a command.



1. **Fg-** The "fg" command in Unix-like systems brings a suspended or backgrounded process to the foreground, allowing the user to interact with it. When a process is in the foreground, it receives input from the user and sends output to the terminal. Processes can be suspended using the "bg" or "Ctrl-Z" commands, and the "fg" command can be used with an optional job ID argument to bring a specific process to the foreground. Once the "fg" command is executed, the specified process resumes and is brought to the foreground.



1. **bg-** On Unix-like operating systems, bg is a job control command. It resumes suspended jobs in the background, returning the user to the shell prompt while the job runs. The presence of bg is required for a shell to comply with the POSIX standard.



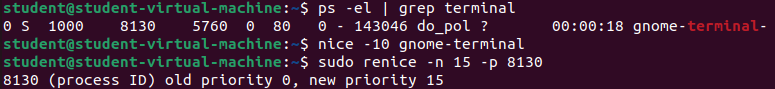
1. **pgrep-** pgrep looks through the currently running processes and lists the process IDs which match the selection criteria to stdout.





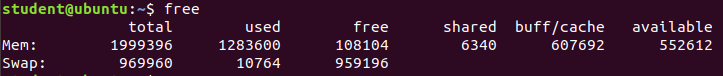


1. **renice-** The renice command allows you to change and modify the scheduling priority of an already running process.

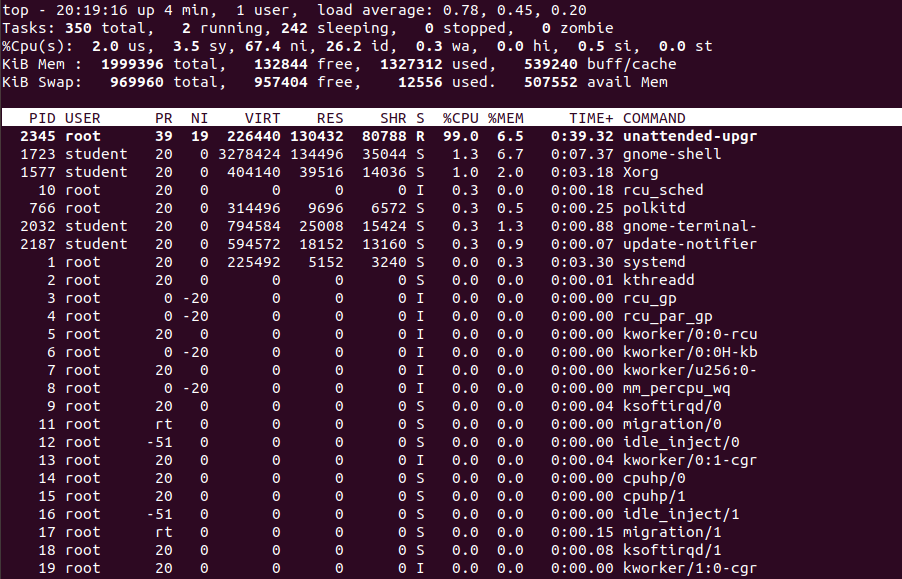


***B) Memory Management***

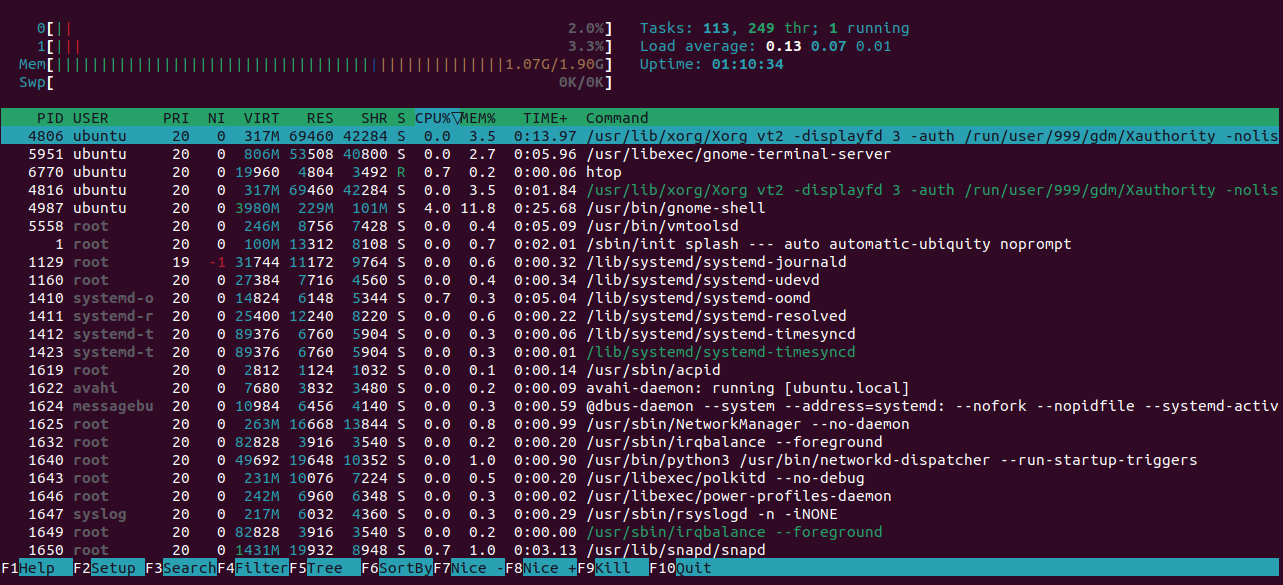
1. **free-** The free command is used to display the total amount of free and used memory in the system, as well as the amount of memory used by the kernel. It also provides information about the swap space usage in the system.



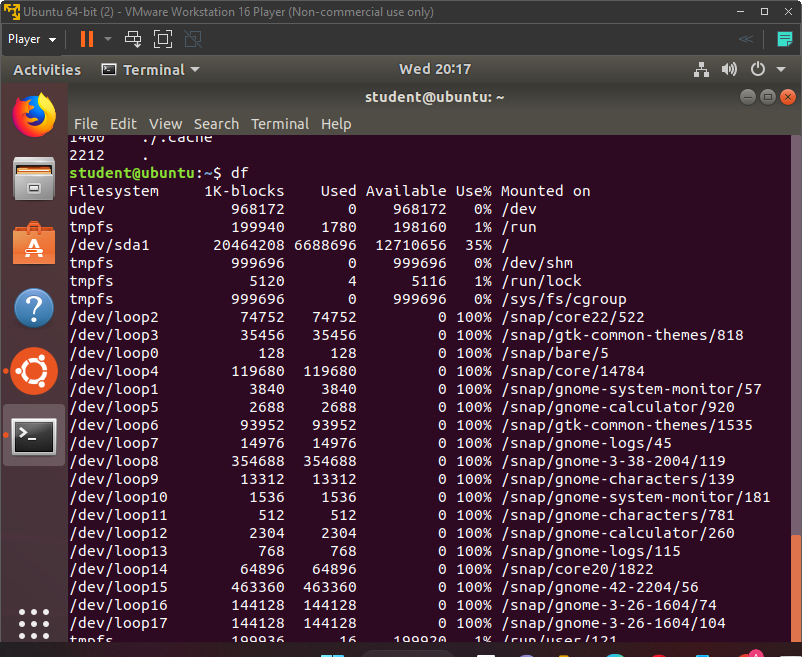
1. **top-** top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.



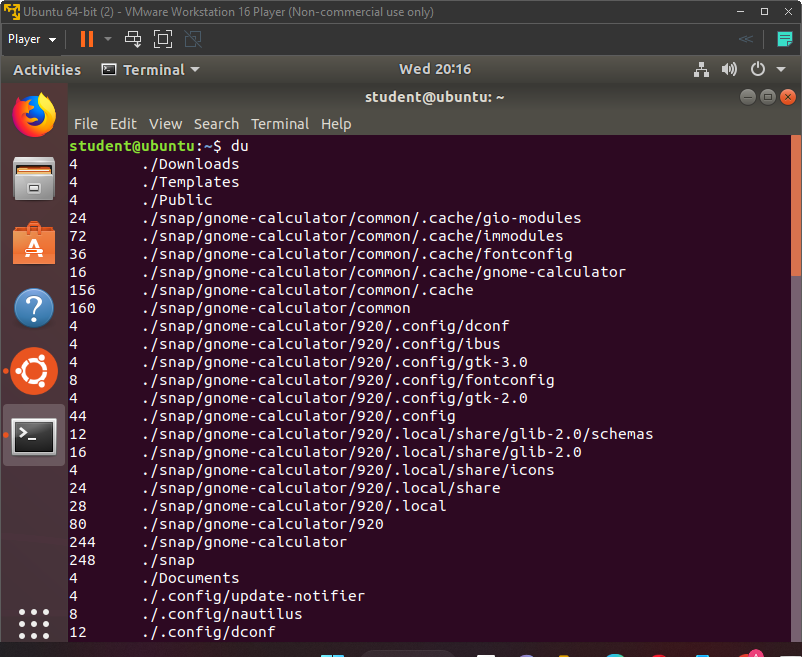
1. **htop-** htop command in Linux system is a command line utility that allows the user to interactively monitor the system’s vital resources or server’s processes in real time. *htop* is a newer program compared to [top](https://www.geeksforgeeks.org/top-command-in-linux-with-examples/) command, and it offers many improvements over top command.



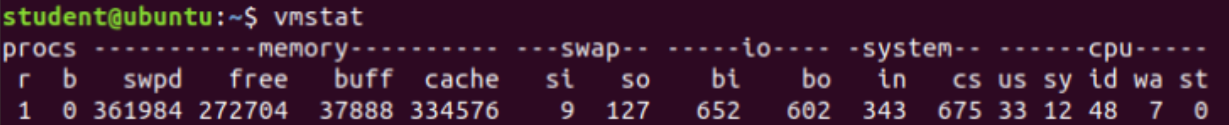
1. **df-** The df command (short for disk free), is used to display information related to file systems about total space and available space.



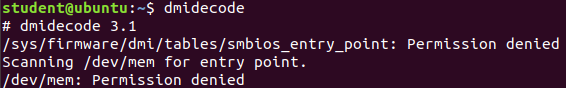
1. **du-** du command, short for disk usage, is used to estimate file space usage.The du command can be used to track the files and directories which are consuming excessive amount of space on hard disk drive.



1. **vmstat-** *vmstat* command in Linux/Unix is a *performance monitoring command* of the system as it gives the information about processes, memory, paging, block IO, disk, and CPU scheduling. All these functionalities makes the command *vmstat* also known as *virtual memory statistic reporter*.



1. **dmidecode-** *dmidecode* also referred as Desktop Management Interface table decoder, record data from DMI table and produce it in human readable format. *dmidecode* command is used when the user want to retrieve system’s hardware related information such as Processor, RAM(DIMMs), BIOS detail, Memory, Serial numbers etc.



1. **pagesize-** The pagesize command prints the size, in bytes, of a page of memory, as returned by the getpagesize subroutine.



**Experiment No. 5 :*****Basic Scripts***

***b) (i) Write a shell script to perform arithmetic operations.***

$ x=8

y=2

echo "x=8, y=2"

echo "Addition of x & y"

echo $(( $x + $y ))

echo "Subtraction of x & y"

echo $(( $x - $y ))

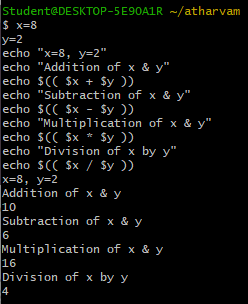
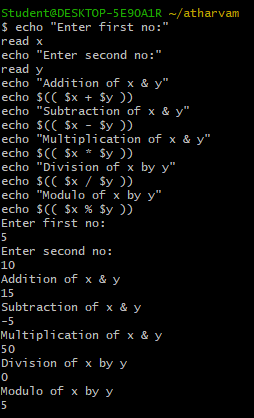
echo "Multiplication of x & y"

echo $(( $x \* $y ))

echo "Division of x by y"

echo $(( $x / $y ))

**OUTPUT-**

** **

***ii) Write a shell script to calculate simple interest.***

echo "Enter Principle Amount:"

read p

echo "Enter Time period:"

read t

echo "Enter rate of interest:"

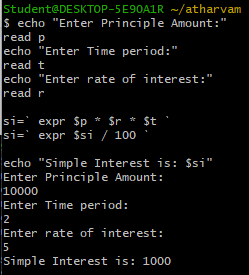
read r

si=` expr $p \\* $r \\* $t `

si=` expr $si / 100 `

echo "Simple Interest is: $si"

**OUTPUT-**

****

***iii) Write a shell script to determine largest among three integer numbers.***

***3) if (a>b) ? (a>c ? a : c) : (b>c ? b :c)***

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo $num1 is largest

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

then

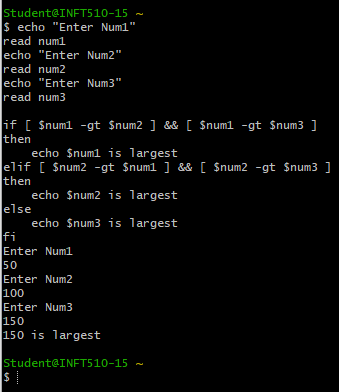
echo $num2 is largest

else

echo $num3 is largest

fi

**OUTPUT-**

****

***(iv) Write a shell script to determine a given year is leap year or not.***

echo "TO FIND LEAP YEAR"

echo -n "Enter a year:"

read year\_checker

if [ `expr $year\_checker % 4` -eq 0 ]

then

echo "$year\_checker is a leap year"

else

echo "$year\_checker is not a leap year"

fi

**OR**

#!/bin/bash

# Prompt user for a year

echo "Enter a year to check if it is a leap year:"

read year

# Check if the year is a leap year

if [[ $((year % 4)) -eq 0 && ($((year % 100)) -ne 0 || $((year % 400)) -eq 0) ]]

then

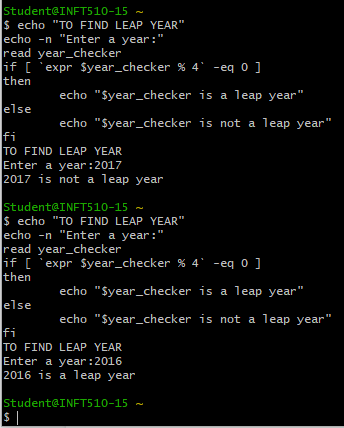
echo "$year is a leap year."

else

echo "$year is not a leap year."

fi

**OUTPUT-**

****

***v) Write a shell script to print multiplication table of given number using while statement.***

echo "Enter the number -"

read n

i=1

while [ $i -le 10 ]

do

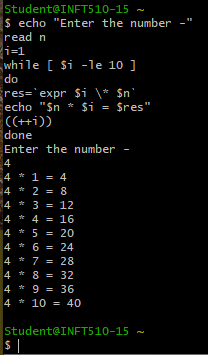
res=`expr $i \\* $n`

echo "$n \* $i = $res"

((++i))

done

**OUTPUT-**

****

**(vi) Write a shell script to search whether element is present is in the list or not.**

#!/bin/bash

# Prompt user to enter a list of elements

echo "Enter a list of elements (separated by spaces):"

read -a elements

# Prompt user to enter a value to search for

echo "Enter a value to search for:"

read value

# Search for value in the list

found=0

for element in "${elements[@]}"

do

if [ "$element" = "$value" ]

then

found=1

break

fi

done

# Print out result

if [ $found -eq 1 ]

then

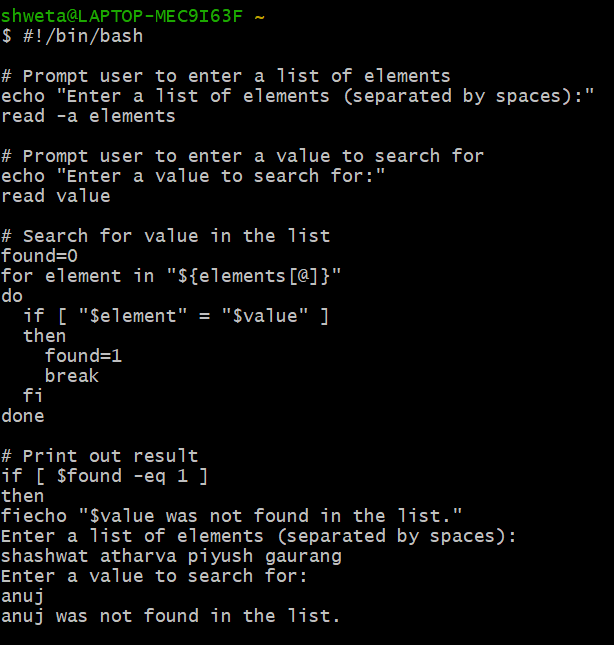
echo "$value was found in the list."

else

echo "$value was not found in the list."

fi

**OUTPUT-**

****

***vii) Write a shell script to compare two strings.***

read -p "Enter two strings: " str1 str2

if [ $str1 == $str2 ]

then

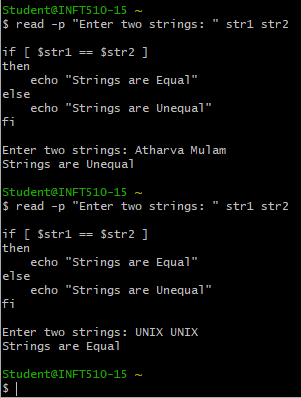
echo "Strings are Equal"

else

echo "Strings are Unequal"

fi

**OUTPUT-**

****

**(viii) Write a shell script to read and check if the directory / file exists or not, if not make the directory / file.**

#!/bin/bash

# Prompt user to enter a file or directory path

echo "Enter a file or directory path:"

read path

# Check if path exists

if [ -e "$path" ]

then

echo "$path already exists."

else

# Check if path is a file or directory

if [ -d "$path" ]

then

# If path is a directory, create it

mkdir "$path"

echo "Directory created: $path"

else

# If path is a file, create it

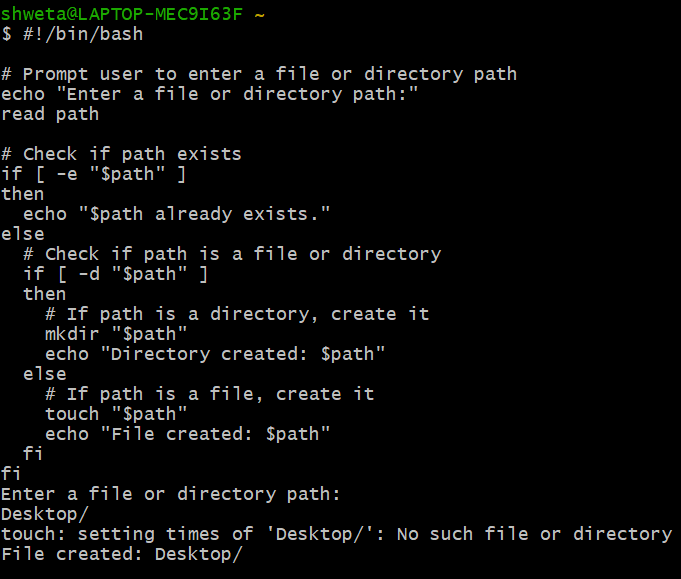
touch "$path"

echo "File created: $path"

fi

fi

**OUPUT-**



**(ix) Write a shell script to implement menu-driven calculator using case statement.**

echo "Enter 1st number:"

read num1

echo "Enter 1st number:"

read num2

echo "Menu:"

echo "1. Add"

echo "2. Subtract"

echo "3. Multiply"

echo "4. Divide"

echo "5. Quit"

while true

do

echo "Enter your choice:"

read choice

case $choice in

1) $(( $num1 + $num2)) ;;

2) $(( $num1 - $num2)) ;;

3) $(( $num1 \* $num2)) ;;

4) $(( $num1 / $num2)) ;;

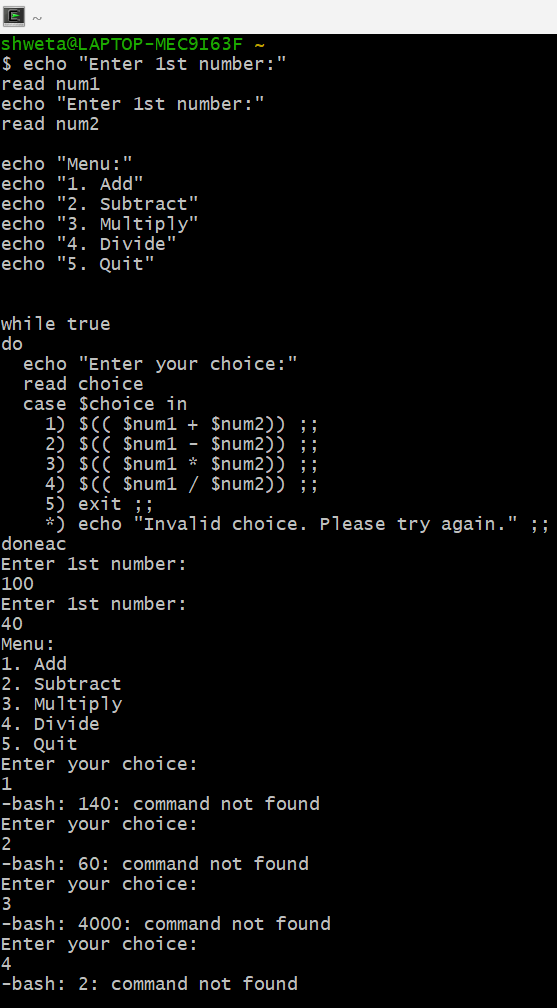
5) exit ;;

\*) echo "Invalid choice. Please try again." ;;

esac

done

**OUTPUT-**



**(x) Write a shell script to print following pattern:**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

#!/bin/bash

for i in {1..4}

do

for j in $(seq 1 $i)

do

echo -n "\* "

done

echo ""

done

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**Experiment No. 6 :*****Advanced Scripts***

***a) Execute the following scripts using grep / sed commands:***

***(i) Write a script using grep command to find the number of words character, words and lines in a file.***

***Process***

#First create directory using mkdir command

#Then cd to that directory

#Then touch wordcount

#Then navigate to C:\cygwin64\home\Student\atharvam from file explorer

#Then open the file which is present there wordcount and write the text whose no. of letters is to #be counted

#Then save it as wordcount.sh

#Then go to cgwin terminal and type chmod +x wordcount.sh

#Then type the following code on cgwin terminal:

**CODE-**

#!/bin/bash

# Prompt the user for the file path

echo "Enter the path to the file:"

read filepath

# Check if the file exists

if [ ! -f "$filepath" ]; then

echo "Error: File not found."

exit 1

fi

# Use grep to count the number of words, characters, and lines in the file

wordcount=$(grep -o '\w\+' "$filepath" | wc -w)

charcount=$(grep -o . "$filepath" | wc -l)

linecount=$(grep -c ^ "$filepath")

# Print the results

echo "Word count: $wordcount"

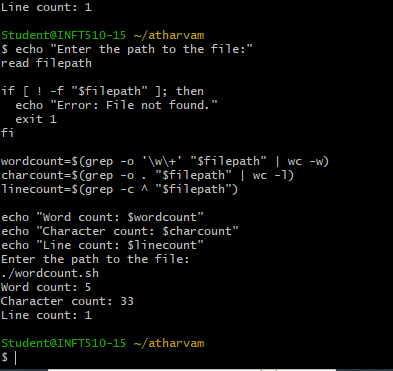
echo "Character count: $charcount"

echo "Line count: $linecount"

Then type :-

./wordcount.sh

**OUTPUT-**



**(ii) Write a script using egrep command to display list of specific type of files in the directory.**

#!/bin/bash

echo "Enter directory path:"

read directory

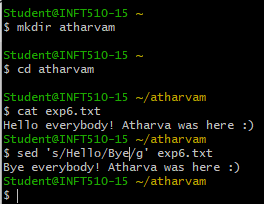
echo "Enter file type:"

read filetype

echo "List of $filetype files in $directory:"

egrep -i "\.$filetype$" $directory/\*

***(iii) Write a script using sed command to replace all occurrences of particular word in a given file.***

****

**(iv) Write a script using sedcommand to print duplicated lines in input.**

#!/bin/bash

echo "Enter input:"

read input

echo "Duplicated lines in input:"

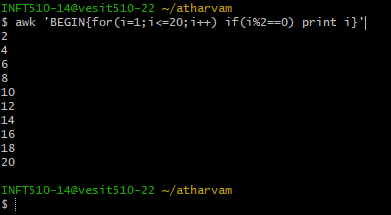
echo "$input" | sed -n 'G; s/\n/&&/; /^\([ -~]\*\n\).\*\n\1/d; s/\n//; h; P'

***b) Execute the following scripts using awk / perl languages:***

***(i) Write an awk script to print all even numbers in a given range.***

**awk '{for(i=$1;i<=$2;i++) if(i%2==0) print i}' ~/data/input.txt**

awk 'BEGIN{for(i=10;i<=20;i++) if(i%2==0) print i}' Ye chal hai but hc hai

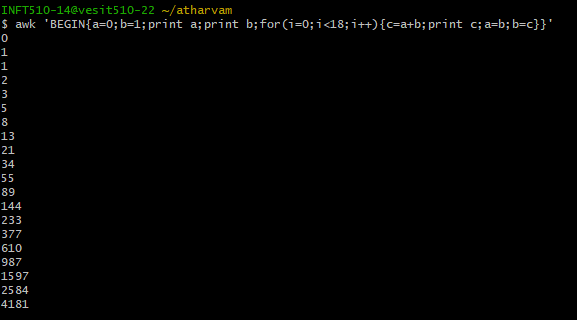
****

***(ii) Write an awk script to develop a Fibonacci series (take user input for number of terms).***

***CODE-***

awk 'BEGIN{a=0;b=1;print a;print b;for(i=0;i<18;i++){c=a+b;print c;a=b;b=c}}'

**OUTPUT-**

****

***(iii) Write a perl script to sort elements of an array.***

**Executed on VMWare workstation-**

# Sorting array in ascending order

@x = sort { $a cmp $b } @fruits;

# Sorting array in descending order

@y = sort { $b cmp $a } @fruits;

# Printing sorted array

print "Array in ascending order: @x\n";

# Printing sorted array

print "Array in descending order: @y\n";

# Initializing an array

@n = (53, 41, 12, 35, 15, 87, 26, 61);

# Printing Original Array

print "Original Array: @n\n";

# Sorting numbers with use of

# spaceship operator

@x = sort { $a <=> $b } @n;

# Printing sorted array

print "Array after Sorting: @x\n";

**CODE-**

@numbers = (22,4,65,9,17,96,1);

# Printing Original Array

print "Original Array: @numbers\n";

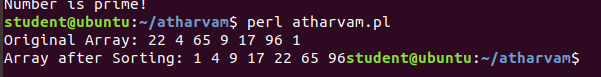
# Sorting numbers with use of spaceship operator

@x = sort { $a <=> $b } @numbers;

# Printing sorted array

print "Array after Sorting: @x";

**OUTPUT-**

****

***(iv) Write a perl script to check a number is prime or not.***

***CODE-***

print "Enter a number: ";

$n=<>;

$d=0;

if($n==2)

{

print "Prime number.n";

}

else

{

for($c=2;$c<=$n-1;$c++)

{

if($n%$c==0)

{

$d=1;

break;

}

}

if($d==1)

{

print "Number is not prime!\n";

}

else

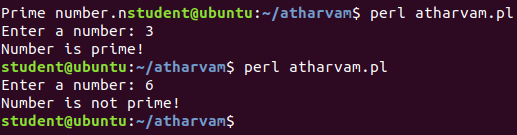
{

print "Number is prime!\n";

}

}

**OUTPUT-**

****