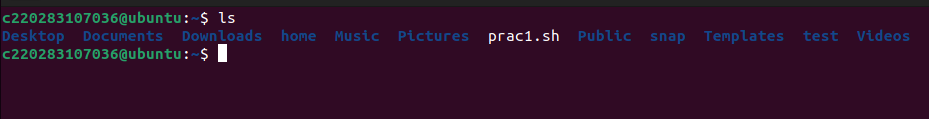
**Practical 1**

**AIM : General Practical.**

1. **Study of Basic Commands of Linux/UNIX.**

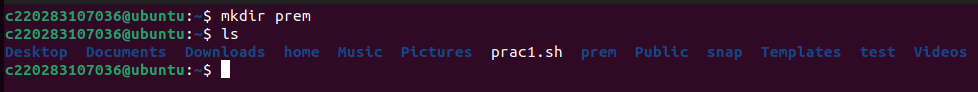
# ls commands:

The ls is the list command in Linux. It will show the full list or content of your directory. Just type ls and press the enter key. The whole content will be shown.

****

# mkdir commands:

The mkdir stands for 'make directory'. With the help of mkdir command, you can create a new directory wherever you want in your system. Just type "mkdir <dir name> , in place of <dir name> type the name of new directory, you want to create and then press enter.

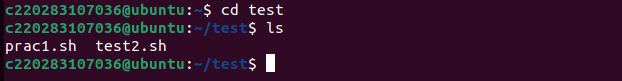


* **cd commands:**

Linux cd command is used to change the current working directory (i.e., in which the current user is working). The "cd" stands for 'change directory.' It is one of the most frequently used commands in the Linux terminal.

Syntax: cd <dirname>

With the help of this command, we can move all over our directories in our system. We can go to our previous directory or previous to the next directory, or anywhere.



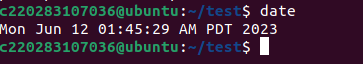
# pwd commands:

PWD stands for Print Working Directory. It writes the complete path name of the working directory to standard output in UNIX-like and other operating systems.



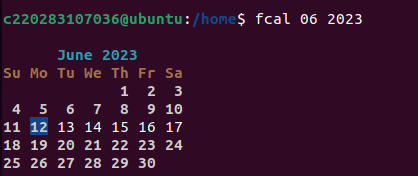
# date commands:

Linux date command is used to display date, time, time zone, etc. It is also used to set the date and time of the Linux system. Generally, it is used to display the date in different formats and calculate dates over time.



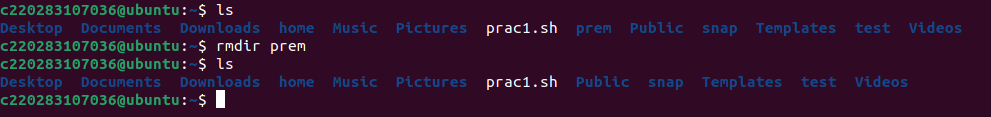
# cal commands:

The 'cal' term stands for calender. It displays current month's calender with current day highlighted.



# rmdir commands:

This command is used to delete a directory. But will not be able to delete a directory including a sub-directory. It means, a directory has to be empty to be deleted.



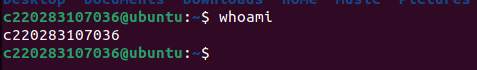
# Shutdown commands:

The shutdown command brings the system down in a secure way. When the shutdown is initiated, all logged-in users and processes are notified that the system is going down, and no further logins are allowed. You can shut down your system immediately or at the specified time.

# whoami commands:

whoami command is used both in Unix Operating System and as well as in Windows Operating System.

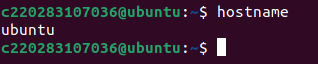
It is basically the concatenation of the strings “who”,”am”,”i” as whoami. It displays the username of the current user when this command is invoked. It is similar as running the id command with the options -un.



# hostname commands:

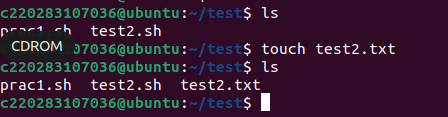
Linux hostname command allows us to set and view the hostname of the system. A hostname is the name of any computer that is connected to a network that is uniquely identified over a network. It can be accessed without using a particular IP address.

By default, the hostname of a system is set during the installation of OS. Even if we install a virtual machine, it is dynamically assigned by the system. But, there may be some conditions whenever we want to change the hostname.



# touch commands:

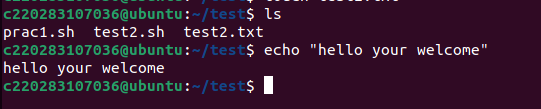
Touch command is a way to create empty files. You can update the modification and access time of each file with the help of touch command.



# echo commands:

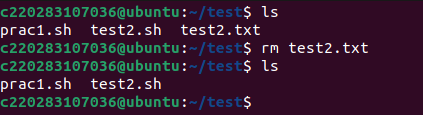
In Linux, the echo command can be used for displaying a line of string/text that is passed as the arguments. This command is a built-in that is mostly and widely used in various batch files and shell scripts to outcome status test to a file and screen.

Here, in 1st snapshot echo is used to create a new file in directory. While in 2nd snapshot echo used with some argument which was add on that particular file and it is displayed by cat command.



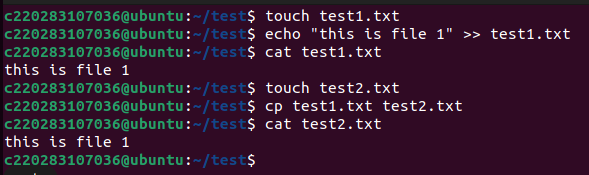
# rm commands:

The 'rm' means remove. This command is used to remove a file. The command line doesn't have a recycle bin or trash unlike other GUI's to recover the files. Hence, be very much careful while using this command. Once you have deleted a file, it is removed permanently.



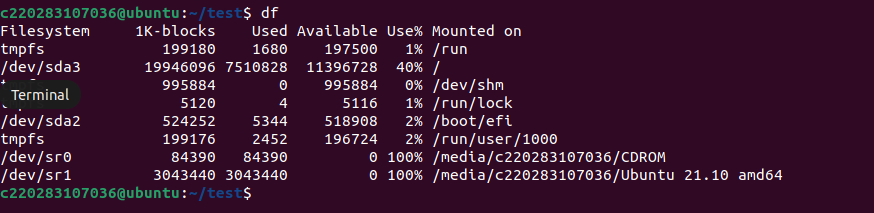
# cp commands:

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



* **df commands:**

the df command is used to display the free disc space of a specific file system. Simply type **"df"** to see a summary of the filesystem's information.



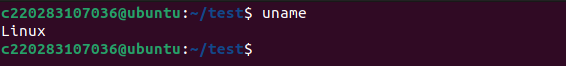
* **Less commands:**

**less** displays the output in the default format and assumes default behavior. The options modify the output or change how **less** acts while processing the file.



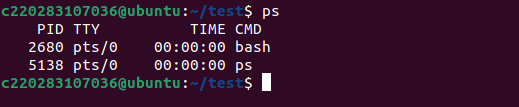
* **Uname Command:**

The **uname** command writes to standard output the name of the operating system that you are using.

****

* **Ps command:**

The ps command, short for Process Status, is a command line utility that is used to **display or view information related to the processes running in a Linux system**. As we all know, Linux is a multitasking and multiprocessing system. Therefore, multiple processes can run concurrently without affecting each other. The ps command lists current running processes alongside their PIDs and other attributes.



1. **Study of unix shell and environment variables.**

Environment variables or ENVs basically define the behavior of the environment. They can affect the processes ongoing or the programs that are executed in the environment.

Scope of an environment variable

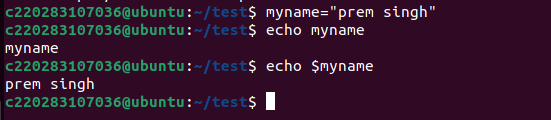
Scope of any variable is the region from which it can be accessed or over which it is defined. An environment variable in Linux can have global or local scope.

Global

A globally scoped ENV that is defined in a terminal can be accessed from anywhere in that particular environment which exists in the terminal. That means it can be used in all kind of scripts, programs or processes running in the environment bound by that terminal.

Local

A locally scoped ENV that is defined in a terminal cannot be accessed by any program or process running in the terminal. It can only be accessed by the terminal( in which it was defined) itself.

****

1. **Write an awk program using function, which convert each word in a given text into capital.**

echo "Enter the String"

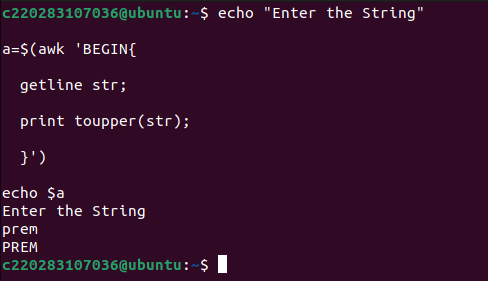
a=$(awk 'BEGIN{

getline str;

print toupper(str);

}')

echo $a

****

**Practical 2**

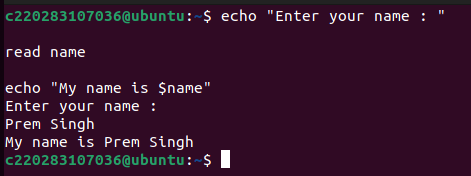
**AIM : Shell Programs**

1. **Write a shell script to display your name**.

echo "Enter your name : "

read name

echo "My name is $name"



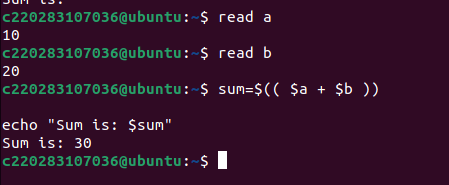
1. **Write a shell script to accept two numbers from user and display addition of it.**

a=10

b=20

sum=$(( $a + $b ))

echo "Sum is: $sum"



1. **Write a shell script to generate mark sheet of a student.**

**Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student**.

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "Student Marksheet"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "Enter Operating System Marks:"

read os

echo "Enter C++ Marks:"

read cpp

echo "Enter Java Marks:"

read java

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

total=`expr $os + $cpp + $java`

echo "Total Marks:"$total

percentage=`expr $total / 3`

echo "Percentage:" $percentage %

if [ $percentage -ge 60 ]

then

echo "Class: First Class Distinction"

elif [ $percentage -ge 50 ]

then

echo "Class: First class"

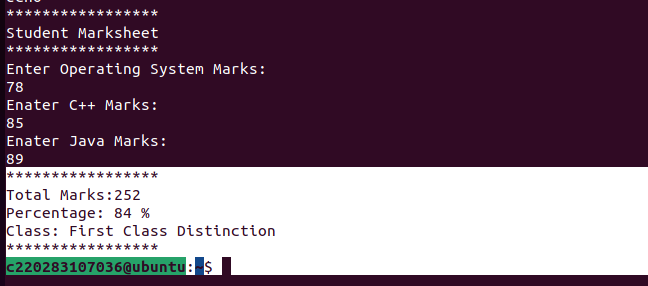
elif [ $percentage -ge 40 ]

then

echo "Class: Second class"

else

echo "Class: Fail"

****

1. **Write a shell script to check entered string is palindrome or not.**

echo "Enter a String"

read input

reverse=""

len=${#input}

for (( i=$len-1; i>=0; i-- ))

do

reverse="$reverse${input:$i:1}"

done

if [ $input == $reverse ]

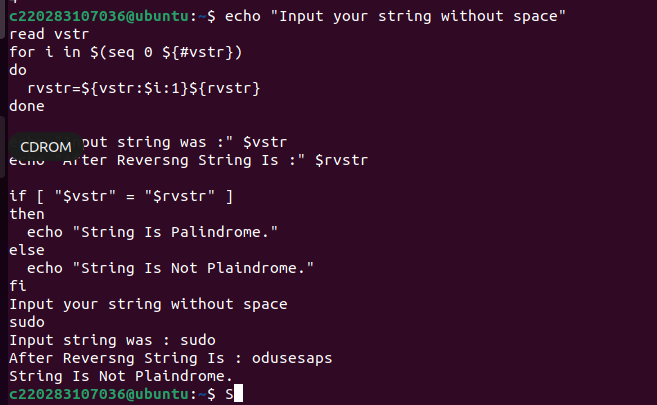
then

echo "$input is palindrome"

else

echo "$input is not palindrome"

fi



1. **Write a shell script which will accept a number b and display first n prime numbers as output.**

echo -e "Enter Number : \c"

read n

for((i=2; i<=$n/2; i++))

do

ans=$(( n%i ))

if [ $ans -eq 0 ]

then

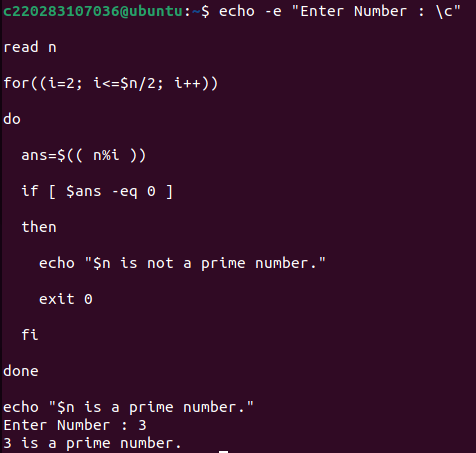
echo "$n is not a prime number."

exit 0

fi

done

echo "$n is a prime number."

****

1. **Write a shell script to illustrate switch case.**

$ read -p "Enter the fruit : " fruit

Enter the fruit : orange

$ case "$fruit" in "apple")

echo "You selected Apple Fruit";;

"banana")

echo "You selected banana fruit";;

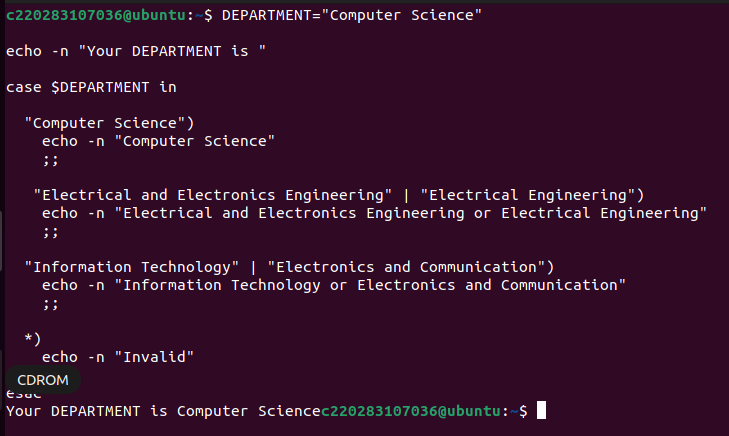
"orange")

echo "you selected orange fruit";;

\*)

echo"Unkonw fruit";;

esac

****

1. **Write a shell script to find factorial of given number n.**

echo "Enter a number"

# Read the number

read num

fa

ct=1

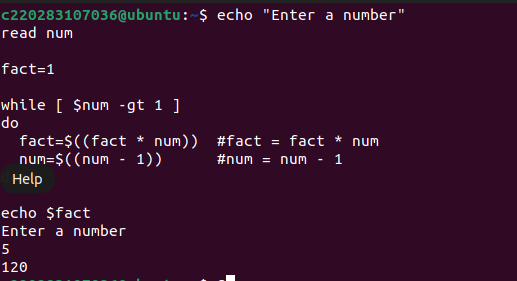
for((i=2;i<=num;i++))

{

fact=$((fact \* i))

}

echo $fact

****

1. **Write a shell script which will accept a number b and display first n prime numbers as output.**

read -p "Enter the number: " n

echo "The prime numbers $n are: "

m=2

while [ $m -le $n ]

do

i=2

flag=0

while [ $i -le `expr $m / 2` ]

do

if [ `expr $m % $i` -eq 0 ]

then

flag=1

break

fi

i=`expr $i + 1`

done

if [ $flag -eq 0 ]

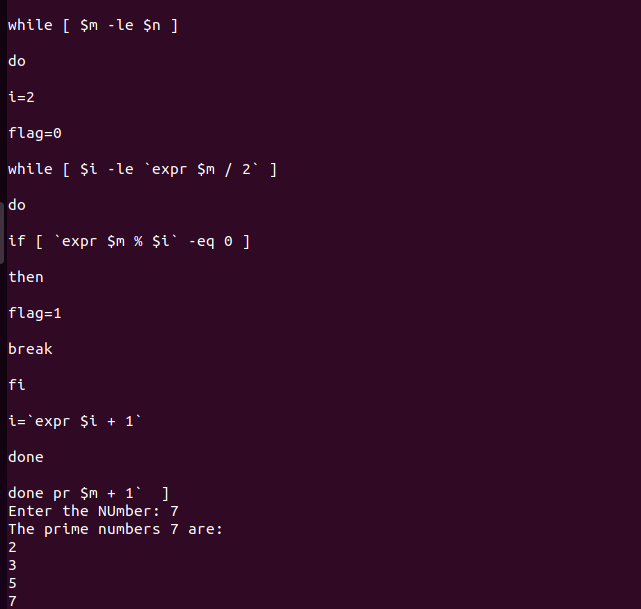
then

echo $m

fi

m=`expr $m + 1`

done

****

1. **Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3, 5, 13, …**

read -p "Enter the Number: " number

x=0

y=1

i=2

echo "Fibonacci Series Upto $number Number: "

echo "$x"

echo "$y"

while [ $i -lt $number ]

do

i=`expr $i + 1`

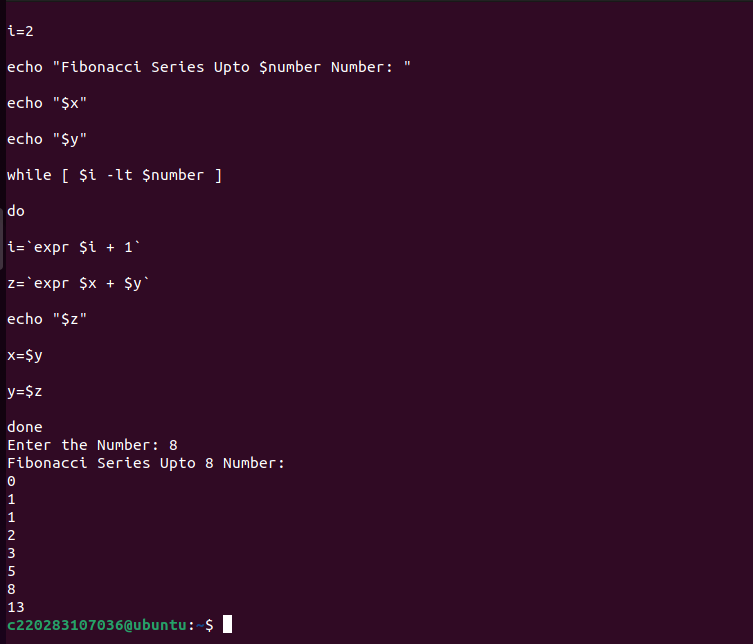
z=`expr $x + $y`

echo "$z"

x=$y

y=$z

done

****

1. **Write a shell script to read n numbers as command arguments and sort them in descending order.**

read -p "Enter The Number: " n

for((i=0; i<$n; i++))

do

read -p "Enter value of arr[$i]: " arr[$i]

done

#sorting code

for((i=0; i<$n; i++))

do

for((j=0; j<n-i-1; j++))

do

if [ ${arr[j]} -lt ${arr[$((j+1))]} ]

then

#swapping

temp=${arr[j]}

arr[$j]=${arr[$((j+1))]}

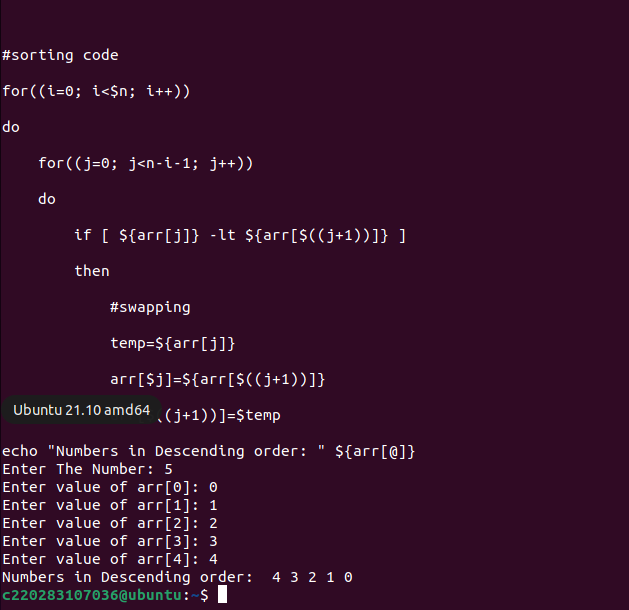
arr[$((j+1))]=$temp

fi

done

done

echo "Numbers in Descending order: " ${arr[@]}

****

1. **Write shell script to display all executable files, directories and zero sized files from current directory.**

echo Executable files

files=$(find lab\_solutions -executable -type f)

echo $files

echo

echo List of Directories

dir=$(ls -d )

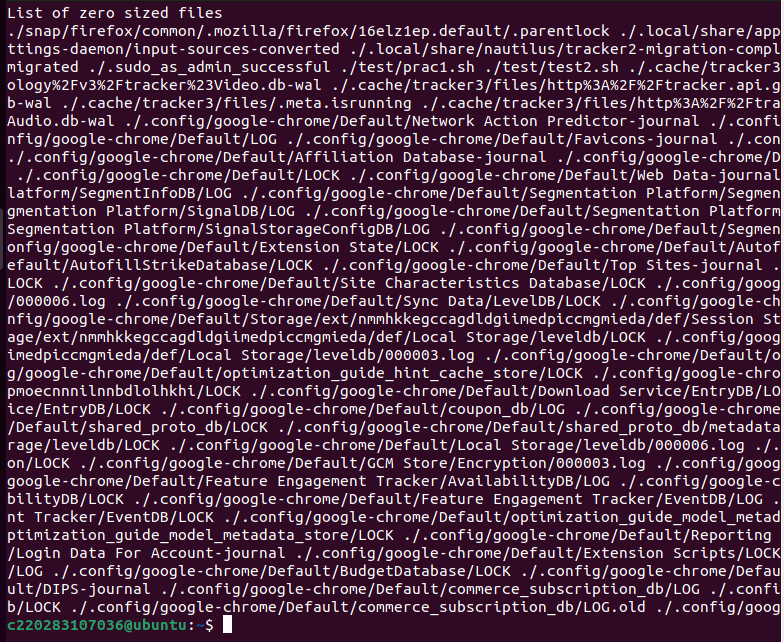
echo $dir

echo

echo List of zero sized files

zero=$(find -size 0)

echo $zero

****

1. **Write a shell script to input numbers and perform addition, subtraction, division and multiplication on that numbers.**

echo "Enter the first number"

read a

echo "Enter the second number"

read b

echo "Enter the operator:"

echo -e "Addition: +\nSubtraction: -\nMultiplication: x\nDivision: /"

read op

**case** $op **in**

+) c=`expr $a + $b`

echo "Sum of $a and $b is $c";;

-) c=`expr $a - $b`

echo "Difference of $a and $b is $c";;

x) c=`expr $a \\* $b`

echo "Product of $a and $b is $c";;

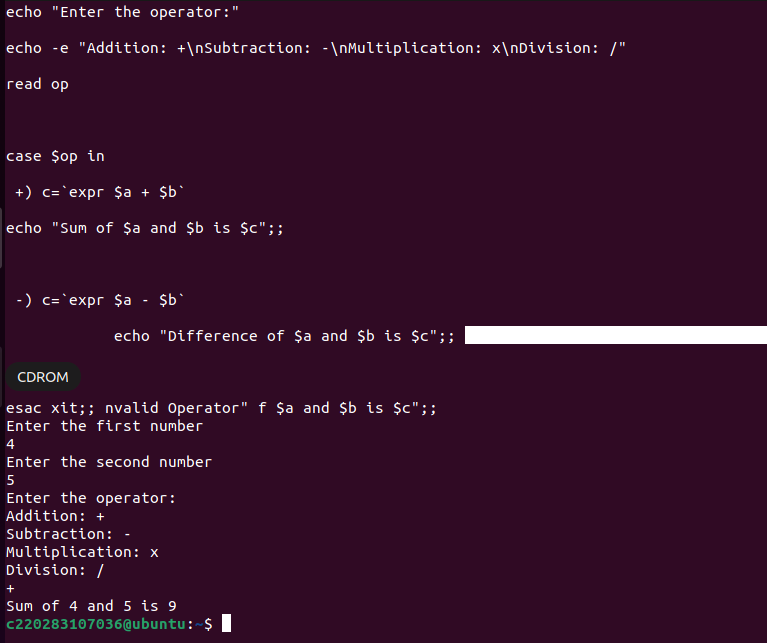
/) c=`expr $a / $b`

echo "Division of $a and $b is $c";;

\*) echo "Invalid Operator"

exit;;

**esac**

****

**13. Write a shell script to find the largest number among of three numbers.**

echo "Enter first number: "

read num1

echo "Enter second number: "

read num2

echo "Enter third number: "

read num3

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

then

echo "$num1 is the largest number."

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

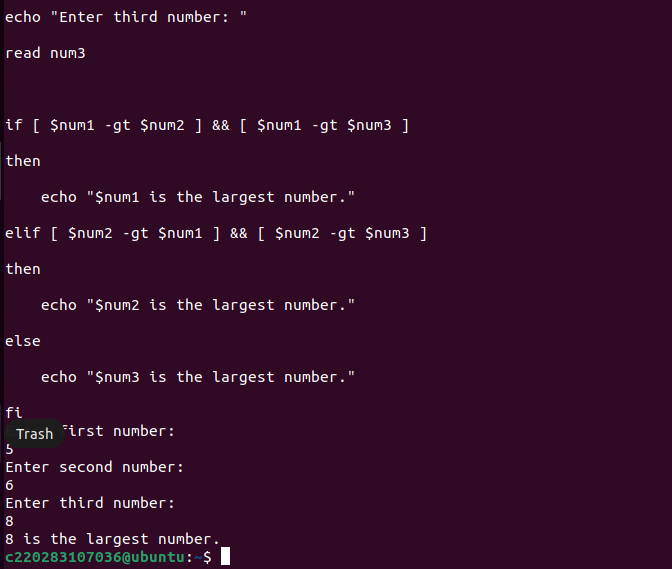
then

echo "$num2 is the largest number."

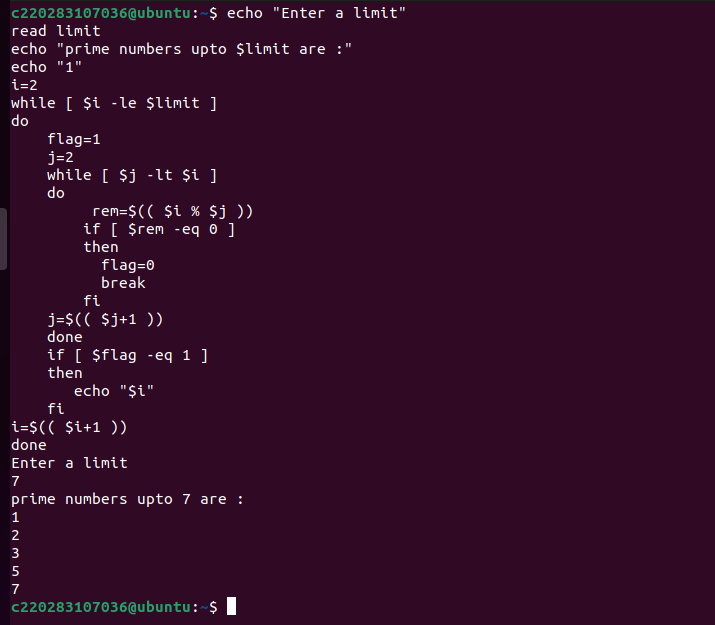
else

echo "$num3 is the largest number."

fi

****

**14. Write a shell script to generate prime numbers from 1 to n, where n is any positive integer entered by user.**

****

**16. write a shell script for converting a decimal to binary, octal, hexadecimal equivalent.**

echo "Enter a decimal number:"

read n

x=$n

y=$n

z=$n

binary(){

i=1

ans=0

a=2

while [ $x -ne 0 ]

do

ans=$(($ans + ($x%$a)\*$i))

x=$(($x/$a))

i=$(($i\*10))

done

echo "binary number is : $ans"

}

octal(){

i=1

ans=0

a=8

while [ $y -ne 0 ]

do

ans=$(($ans + ($y%$a)\*$i))

y=$(($y/$a))

i=$(($i\*10))

done

echo "octal number is : $ans"

}

hexadecimal(){

i=1

ans=0

a=16

while [ $z -ne 0 ]

do

ans=$(($ans + ($z%$a)\*$i))

z=$(($z/$a))

i=$(($i\*10))

done

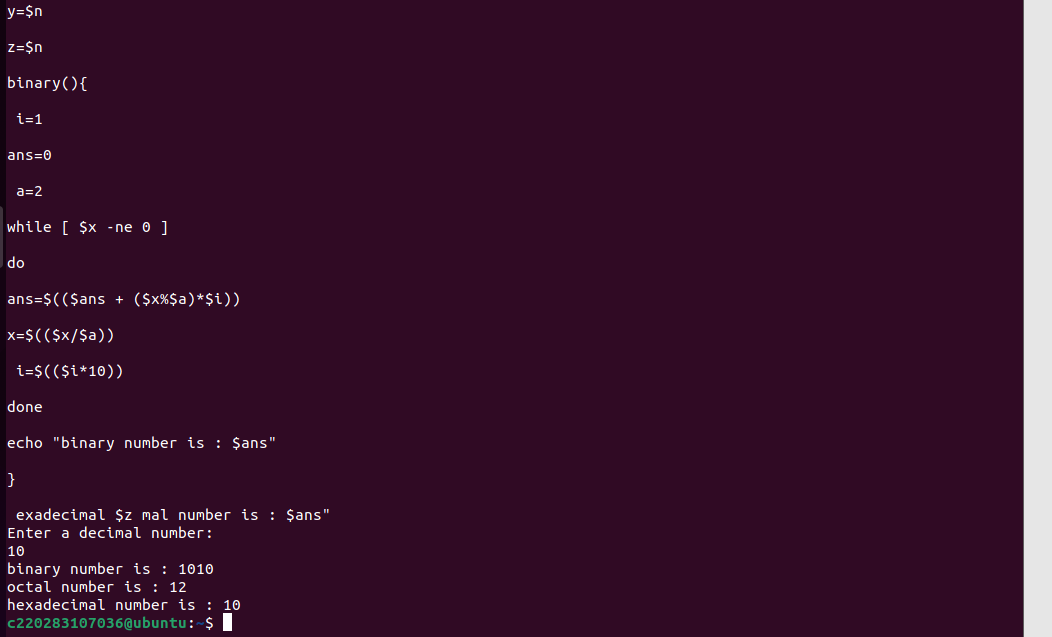
echo "hexadecimal number is : $ans"

}

binary $x

octal $y

hexadecimal $z

****

**17. Write a shell script from addition of digit of given number.**

echo "Enter a Number:"

read n

temp=$n

sd=0

sum=0

while [ $n -gt 0 ]

do

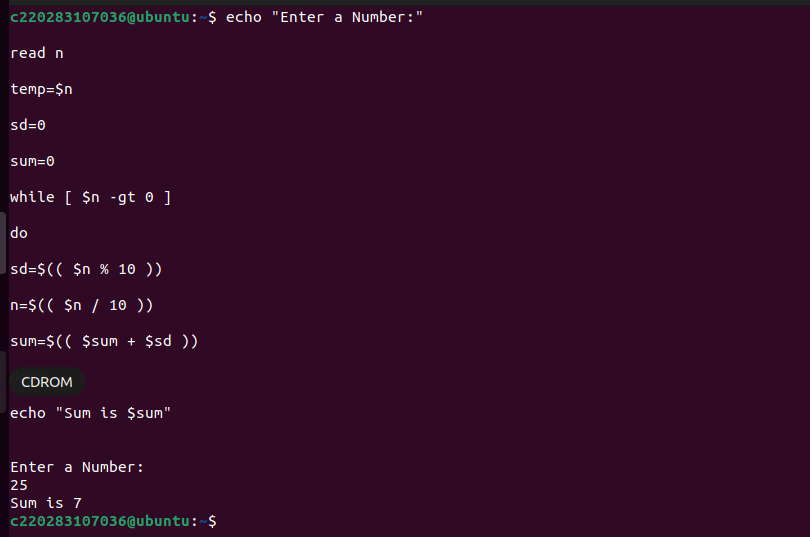
sd=$(( $n % 10 ))

n=$(( $n / 10 ))

sum=$(( $sum + $sd ))

done

echo "Sum is $sum"

****

**18. Write a script which prints good night, good evening, good afternoon and good morning according to system time.**

h=$(date +"%H")

if [ $h -gt 6 -a $h -le 12 ]

then

echo good morning

elif [ $h -gt 12 -a $h -le 16 ]

then

echo good afternoon

elif [ $h -gt 16 -a $h -le 20 ]

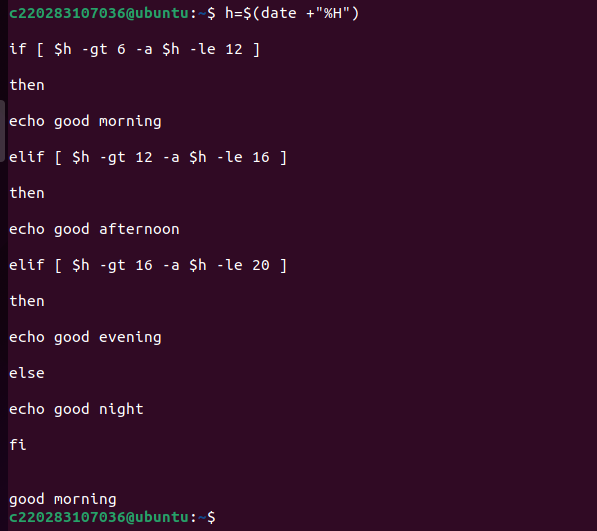
then

echo good evening

else

echo good night

fi

****

**19. Write a shell script to validate the entered date. (eg. Date format is: dd–mm-yyyy)..**

echo "Enter Valid Date"

read date

echo "You have entered $date"

date -d $date

if [ $? -eq 0 ]

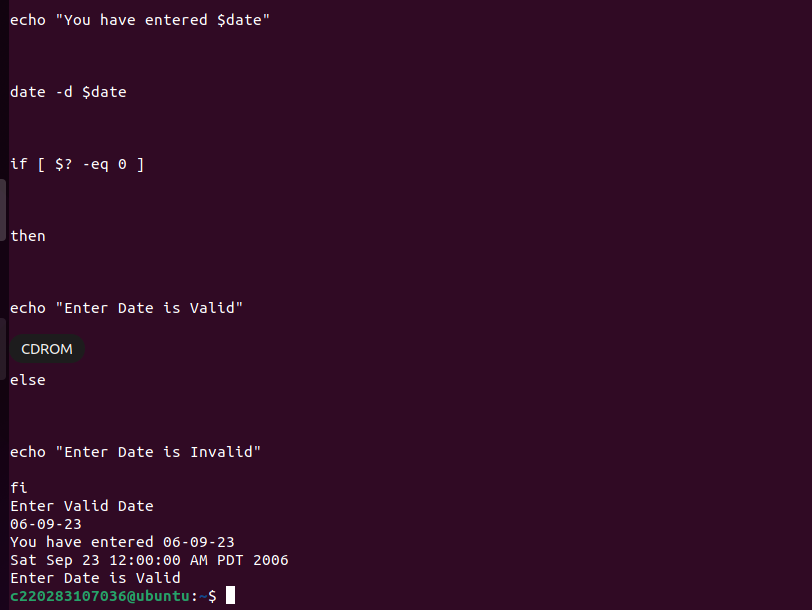
then

echo "Enter Date is Valid"

else

echo "Enter Date is Invalid"

fi

****

**Practical 3**

**AIM : File Practical**

1. **Write a menu driven shell script which will print the following menu and execute the given task.**

* Display calendar of current month
* Display today’s date and time
* Display usernames those are currently logged in the system
* Display your name at given x, y position
* Display your terminal number
* Exit

echo "Select Anyone Option"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "1)Display Calener of current month"

echo "2)Display Today's Date and Time"

echo "3)Display Username who are currently logged in"

echo "4)Display your name on given x,y position"

echo "5)Display your terminal Number"

echo "6)Exit"

echo "Enter your choice:"

read ch

case $ch in

1)cal;;

2)date;;

3)who;;

4)row=$(tput lines)

col=$(tput cols)

echo "Terminal Window has Rows=$row Cols=$col"

echo "Enter desired X,Y position"

echo "X position="

read x

echo "Y position="

read y

echo "Enter the name"

read name

tput cup $x $y

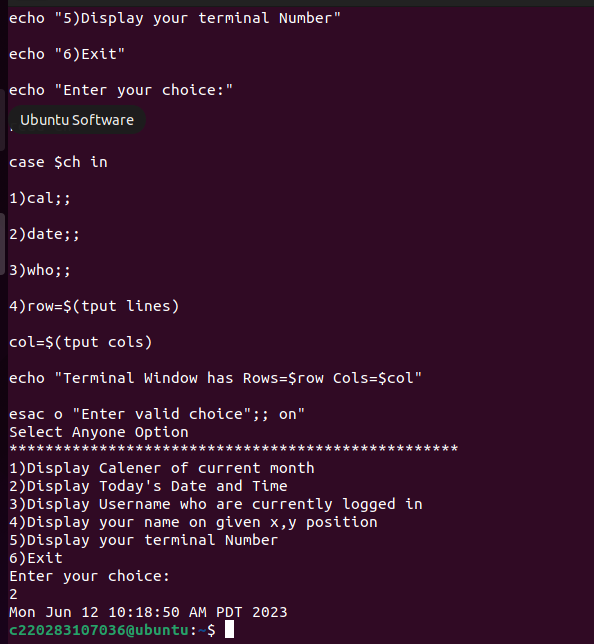
echo "$name";;

5)tty;;

6)echo "Exit";;

\*)echo "Enter valid choice";;

esac

****

**2. Write a shell script to display the menu. It should take appropriate action when an option is selected.**

# List directory.

* Copy file

# Rename a file

* Delete a file

# Edit a file

* Exit

display\_menu() {

echo "Menu:"

echo "1. List directory"

echo "2. Copy file"

echo "3. Rename a file"

echo "4. Delete a file"

echo "5. Edit a file"

echo "6. Exit"

echo -n "Enter your choice: "

}

list\_directory() {

echo "Listing directory..."

ls -l

}

copy\_file() {

echo -n "Enter the source file: "

read source\_file

echo -n "Enter the destination file: "

read destination\_file

cp "$source\_file" "$destination\_file"

echo "File copied successfully!"

}

rename\_file() {

echo -n "Enter the current file name: "

read current\_name

echo -n "Enter the new file name: "

read new\_name

mv "$current\_name" "$new\_name"

echo "File renamed successfully!"

}

delete\_file() {

echo -n "Enter the file name to delete: "

read file\_name

rm "$file\_name"

echo "File deleted successfully!"

}

edit\_file() {

echo -n "Enter the file name to edit: "

read file\_name

# You can replace 'vi' with your preferred text editor

vi "$file\_name"

}

while true; do

display\_menu

read choice

case $choice in

1)

list\_directory

;;

2)

copy\_file

;;

3)

rename\_file

;;

4)

delete\_file

;;

5)

edit\_file

;;

6)

echo "Exiting..."

break

;;

\*)

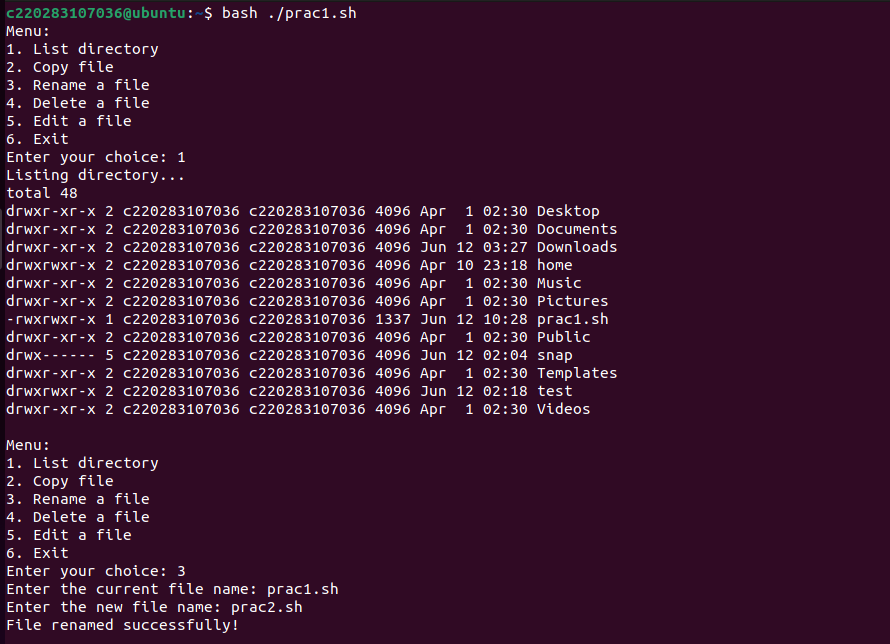
echo "Invalid option! Please try again."

;;

esac

echo

done

****

**3.Write a script to display the files in the current directory in the following fromate.**

**“Filename---------Size In kb---------date---------protection---------owner”**

echo "Filename---------Size In kb---------Date---------Protection---------Owner"

for file in \*; do

if [ -f "$file" ]; then

file\_name=$(basename "$file")

file\_size=$(du -k "$file" | awk '{print $1}')

file\_date=$(date -r "$file" +"%Y-%m-%d")

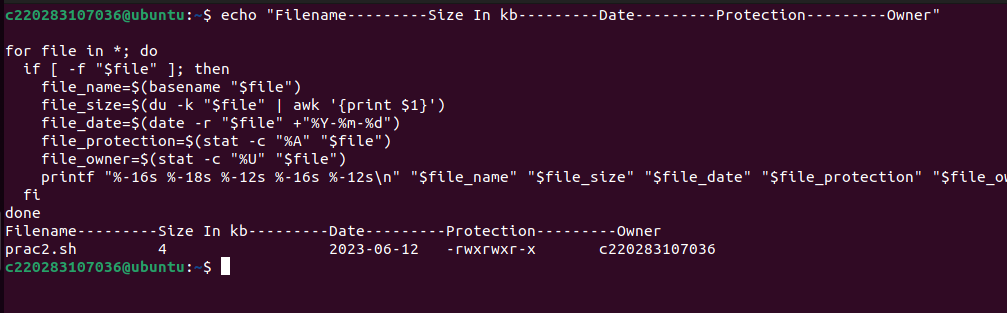
file\_protection=$(stat -c "%A" "$file")

file\_owner=$(stat -c "%U" "$file")

printf "%-16s %-18s %-12s %-16s %-12s\n" "$file\_name" "$file\_size" "$file\_date" "$file\_protection" "$file\_owner"

fi

done

****

**4.Write a script to display the directory in the descending of the size of each file.**

echo "Filename---------Size In kb---------Date---------Protection---------Owner"

# Store file information in an array

file\_info=()

while IFS= read -r -d $'\0' file; do

if [ -f "$file" ]; then

file\_name=$(basename "$file")

file\_size=$(du -k "$file" | awk '{print $1}')

file\_date=$(date -r "$file" +"%Y-%m-%d")

file\_protection=$(stat -c "%A" "$file")

file\_owner=$(stat -c "%U" "$file")

file\_info+=("$file\_name,$file\_size,$file\_date,$file\_protection,$file\_owner")

fi

done < <(find . -maxdepth 1 -type f -print0)

# Sort file information based on size in descending order

IFS=$'\n' sorted\_info=($(sort -t ',' -k2rn <<<"${file\_info[\*]}"))

unset IFS

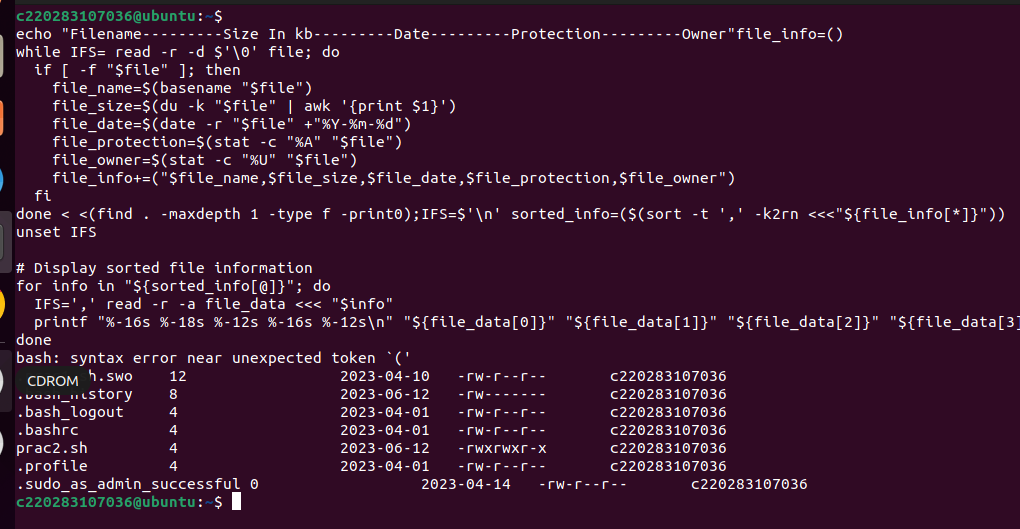
# Display sorted file information

for info in "${sorted\_info[@]}"; do

IFS=',' read -r -a file\_data <<< "$info"

printf "%-16s %-18s %-12s %-16s %-12s\n" "${file\_data[0]}" "${file\_data[1]}" "${file\_data[2]}" "${file\_data[3]}" "${file\_data[4]}"

done

****

**5.Write a shell for generating mark-sheet and calculate grades of the student also.**

calculate\_grade() {

marks=$1

if (( marks >= 90 )); then

echo "A"

elif (( marks >= 80 )); then

echo "B"

elif (( marks >= 70 )); then

echo "C"

elif (( marks >= 60 )); then

echo "D"

else

echo "F"

fi

}

# Function to generate mark sheet for a student

generate\_marksheet() {

echo "Enter student name: "

read name

echo "Enter marks obtained in subject 1: "

read subject1

echo "Enter marks obtained in subject 2: "

read subject2

echo "Enter marks obtained in subject 3: "

read subject3

total\_marks=$((subject1 + subject2 + subject3))

average\_marks=$((total\_marks / 3))

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\* Mark Sheet \*\*\*\*\*\*\*\*\*\*\*\*"

echo "Student Name: $name"

echo "Subject 1: $subject1"

echo "Subject 2: $subject2"

echo "Subject 3: $subject3"

echo "Total Marks: $total\_marks"

echo "Average Marks: $average\_marks"

echo "Grade: $(calculate\_grade $average\_marks)"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

}

# Main menu loop

while true; do

echo "Menu:"

echo "1. Generate Mark Sheet"

echo "2. Exit"

echo -n "Enter your choice: "

read choice

case $choice in

1)

generate\_marksheet

;;

2)

echo "Exiting..."

break

;;

\*)

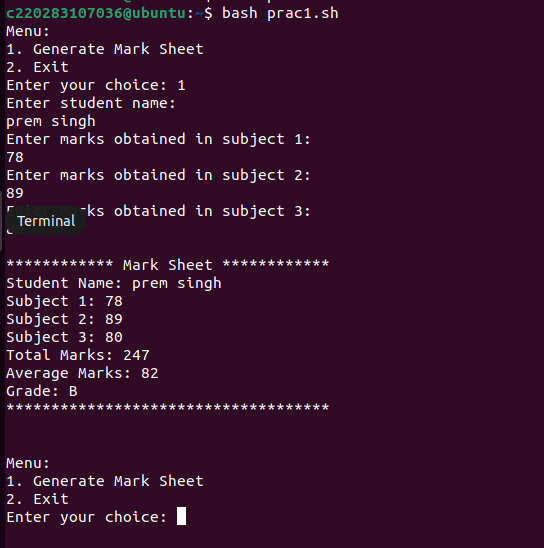
echo "Invalid option! Please try again."

;;

esac

echo

done

****

**6.Write a shell script to display the following menu, it should take appropriate action when an option is selected.**

* **Display current directory.**
* **List directory.**
* **Make Directory.**
* **Change Directory.**
* **Quit.**

display\_menu() {

echo "Menu:"

echo "1. Display current directory"

echo "2. List directory"

echo "3. Make Directory"

echo "4. Change Directory"

echo "5. Quit"

echo -n "Enter your choice: "

}

display\_current\_directory() {

echo "Current Directory: $(pwd)"

}

list\_directory() {

echo "Listing directory..."

ls -l

}

make\_directory() {

echo -n "Enter the directory name: "

read dir\_name

mkdir "$dir\_name"

echo "Directory created successfully!"

}

change\_directory() {

echo -n "Enter the directory path: "

read dir\_path

cd "$dir\_path"

echo "Changed directory to: $(pwd)"

}

while true; do

display\_menu

read choice

case $choice in

1)

display\_current\_directory

;;

2)

list\_directory

;;

3)

make\_directory

;;

4)

change\_directory

;;

5)

echo "Quitting..."

break

;;

\*)

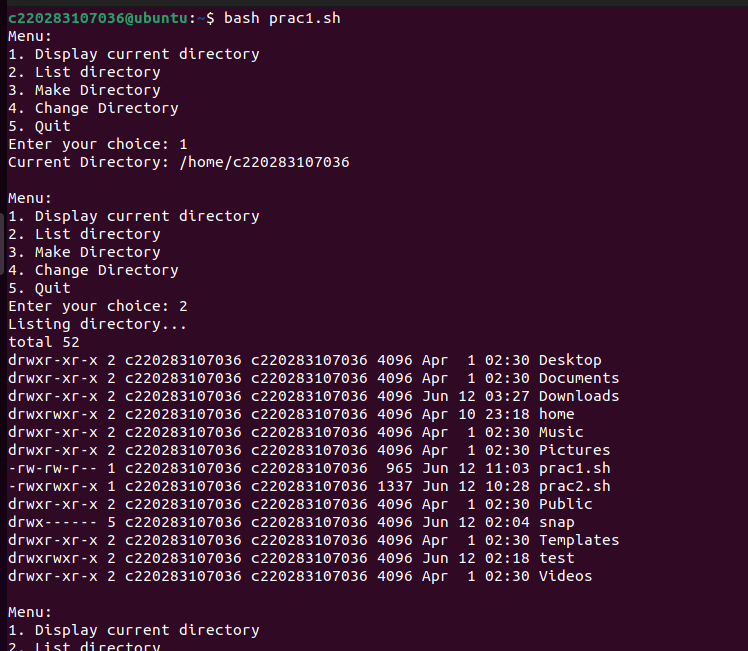
echo "Invalid option! Please try again."

;;

esac

echo

done

****

**7. Write a shell script to implement a simple database system Implement operations like insert record, delete record, view all record, sort records, and query on some field.**

**8.Write aa shell script to display the following menu. It should take appropriate action when an option is selected.**

* **Display names of all users currently logged in.**
* **Display name of all members of particular group.**
* **Display name of all users who are in the system.**
* **Quit.**

display\_menu() {

echo "Menu:"

echo "1. Display names of all users currently logged in"

echo "2. Display names of all members of a particular group"

echo "3. Display names of all users who are in the system"

echo "4. Quit"

echo -n "Enter your choice: "

}

display\_logged\_in\_users() {

echo "Users currently logged in:"

who | cut -d' ' -f1 | sort -u

}

display\_group\_members() {

echo -n "Enter the group name: "

read group\_name

echo "Members of group $group\_name:"

getent group "$group\_name" | cut -d: -f4 | tr ',' '\n'

}

display\_all\_users() {

echo "All users in the system:"

getent passwd | cut -d: -f1

}

while true; do

display\_menu

read choice

case $choice in

1)

display\_logged\_in\_users

;;

2)

display\_group\_members

;;

3)

display\_all\_users

;;

4)

echo "Quitting..."

break

;;

\*)

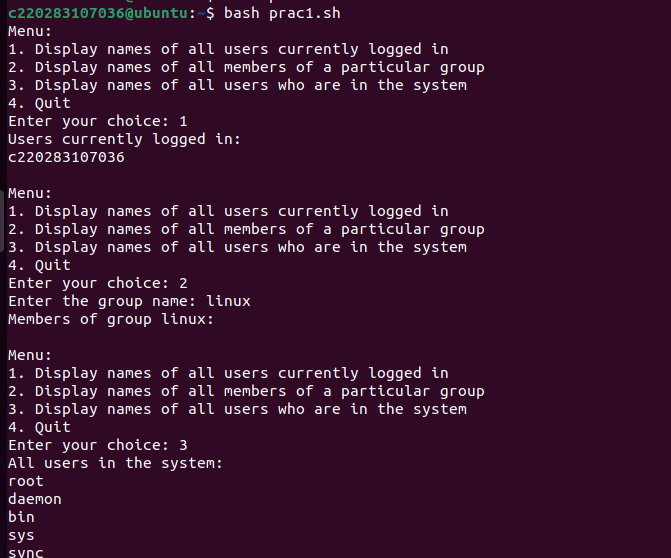
echo "Invalid option! Please try again."

;;

esac

echo

done

****

1. **Write a shell script which prints the following**

* **The message “today is :” with current date in mm/dd/yy format.**
* **The message “no of users logged in :” woth total number of currently logged in users.**
* **The message personal status: “with your own login name, port-number, date and time of login”.**
* **Write a shell script to display the directory in the descending order of the size of each file.**
* **Write a shell script to display the files with zero size and write permission “on”.**

echo "Today is: $(date +%m/%d/%y)"

logged\_in\_users=$(who | wc -l)

echo "No of users logged in: $logged\_in\_users"

login\_name=$(whoami)

port\_number=$(who am i | awk '{print $2}' | sed 's/[()]//g')

login\_date=$(who am i | awk '{print $3}')

login\_time=$(who am i | awk '{print $4}')

echo "Personal status:"

echo "Login name: $login\_name"

echo "Port number: $port\_number"

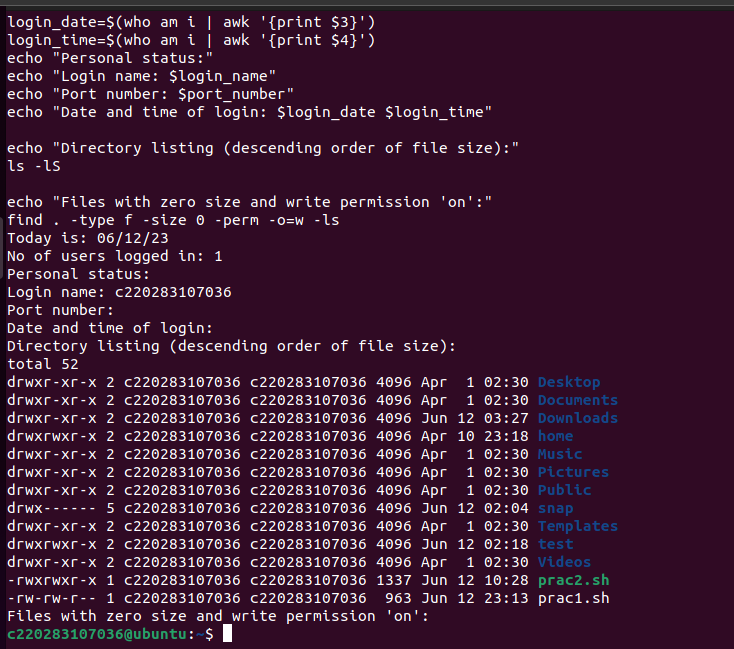
echo "Date and time of login: $login\_date $login\_time"

echo "Directory listing (descending order of file size):"

ls -lS

echo "Files with zero size and write permission 'on':"

find . -type f -size 0 -perm -o=w -ls

****