EX.NO:1 **DIRECTORY MANAGEMENT COMMANDS**

Aim:

To Write a commands to execute the Usage of Directory Management commands: ls, cd, pwd, mkdir, rmdir.

COMMANDS:

1) **mkdir**

Short for make directory this command is used to create a new directory.

Syntax: mkdir dirctoryname

2) **rmdir**

Deletes a directory.

Syntax: rmdir dirctoryname

3) **ls**

Lists the contents of a directory

Syntax: ls option filename

4) **cd**

Changes the directory.

Syntax: cd directory name

5) **pwd(Print working directory)**

Short for print working directory the pwd command displays the name of the current working directory.

Syntax: pwd

Result:

Thus the above commands were executed successfully.

EX.NO:2 **FILE MANAGEMENT COMMANDS**

Aim:

To Write a commands to execute the Usage of File Management commands: cat, chmod, cp, mv, rm, more.

COMMANDS:

1) **cat**

Allows you to look, modify or combine a file.

Syntax: cat filename

2) **chmod**

Changes the permission of a file.

Syntax

chmod [OPTION]... MODE[,MODE]... FILE...

chmod [OPTION]... OCTAL-MODE FILE...

3) **cp**

Copies files from one location to another.

Syntax: cp source destination

4) **rm**

Deletes a file without confirmation (by default).

Syntax: rm filename

5) **mv**

Renames a file or moves it from one directory to another directory.

Syntax: mv oldname newname

6) **more**

Displays text one screen at a time.

Syntax: more

Result:

Thus the above commands were executed successfully.

EX.NO:3 **USE OF GENERAL PURPOSE COMMANDS IN LINUX**

Aim:

Write a Linux commands to Use the General Purpose commands: wc, cal, date, who, tty, ln.

COMMANDS:

1) **who**

Displays who is on the system.

Syntax: who

2) **cal**

Displays calendar for the month and the year.

Syntax: cal year

3) **Date**

Displays the date of that day.

Syntax: date

4) **ln**

Creates a link to a file.

Syntax: ln filename 1 filename 2

5) **tty**

Print the file name of the terminal connected to standard input.

Syntax: tty

6) **wc**

Short for word count, wc displays a count of lines, words, and characters in a file.

Syntax: wc filename

Result:

Thus the above commands were executed successfully.

Ex No: 4 **SIMPLE FILTERS IN LINUX**

Aim:

To write a linux commands to implement simple filters.

Commands:

1) **sort**

To sort the lines of the named files, use sort.

Syntax: sort filename

2) **head**

Displays the first ten lines of a file, unless otherwise stated.

Syntax: head option filename

3) **Tail**

Delivers the last part of the file.

Syntax: tail option filename

4) **paste**

Merge corresponding lines of one or more *files* into vertical columns, separated by a tab.

Syntax: paste filename1 filename2

5) **Pr**

Format one or more *files* according to *options* to standard output. Each page includes a heading that consists of the page number, filename, date, and time.

Syntax: pr filename

6) **nl**

Number the lines of *file* in logical page segments. Numbering resets to 1 at the start of each logical page. Pages

Syntax: nl filename

7) **Cut**

Select a list of columns or fields from one or more *files*. Either -c or -f must be specified. *list* is a sequence of integers

Syntax: cut option filename

Result:

Thus the above commands were executed successfully

EX.NO:5 ADVANCED FILTERS & COMMUNICATION COMMANDS

Aim:

To write a Linux commands to implement advanced filters and communication commands.

ADVANCED FILTERS:

Commands:

1) **egrep**

Search one or more *files* for lines that match a regular expression *regexp*. egrep doesn't support the metacharacters \(, \), *n*, \<, \>, \{, or \}, but does support the other metacharacters, as well as the extended set +, ?, |, and ( ).

Syntax: egrep option filename

2) **grep**

Search one or more *files* for lines that match a regular expression *regexp*.

Syntax: grep option filename.

3) **fgrep**

Search one or more *files* for lines that match a literal, text-string *pattern*.Because fgrep does not support regular expressions, it is faster than grep.

Syntax: fgrep option filename

4) **Uniq**

The **uniq** command in Linux is a command line utility that reports or filters out the repeated lines in a file.

Syntax: uniq option filename

COMMUNICATION COMMANDS:

1) **write**

This is a utility for terminal-to-terminal communication. It allows sending lines from your terminal (console or xterm) to that of another user

Syntax: who

2) **wall**

wall -- To broadcast a message to all users connected to the server. The length of the message is limited to 20 lines

Syntax: wall

Result:

Thus the above commands were executed successfully.

EX.NO:6 PROCESS MANAGEMENT COMMAND IN LINUX

Aim:

To Write a commands to know the details of process status and process management.

COMMANDS:

1) **ps**

Reports the process status.

Syntax: ps

2) **nohup**

Runs a command even if the session is disconnected or the user logs out.

Syntax: nohup.out

3) **kill**

Cancels a job.

Syntax: ps ux

4) **nice**

Invokes a command with an altered scheduling priority.

Syntax: ps –axl

Result:

Thus the above commands were executed successfully

EX.NO:7 DEVICE PATTERN USING META CHARACTER IN LINUX

Aim:

To write a Linux commands to perform device pattern using Meta character to match each of the following situation.

1) All three character filenames.
2) All filenames that contains the characters ‘a ‘or ‘b ‘or’ c.’
3) All filenames beginning with a particular string.
4) All filenames beginning with ‘ca’ and ending with two digits.
5) All filenames beginning with‘s ‘and having ‘a’ at somewhere.

Commands:

1) **ALL THREE CHARACTER FILENAMES.**

Syntax: ls ???

2) **ALL FILENAMES THAT CONTAINS THE CHARACTERS ‘a ‘or ‘b ‘or’ c.’**

Syntax: ls *[abc]*

3) **ALL FILENAMES BEGINNING WITH A PARTICULAR STRING.**

Syntax: ls *mystring* type f

4) **ALL FILENAMES BEGINNING WITH CA AND ENDING WITH TWO DIGITS.**

Syntax: ls ca*[0-9] [0-9]

5) **ALL FILENAMES BEGINNING WITH S AND HAVING ‘A’ AT SOMEWHERE.**

Syntax: ls s*?a*

Result:

Thus the above commands were executed successfully.

Ex No: 8 **DISPLAY THE DECREMENTED VALUE OF N.**

Aim:

To write a Shell script program that accepts a numerical value N & display the decremented value of N till it reaches 0.

Procedure:

1. Create a new vi editor file.
2. Get the integer value of n.
3. Check the n value greater than zero then print the n value.
4. Decrement the variable n.
5. Repeat the steps up to this condition false.(i.e. n>0)
6. Print the result.

Program [root@sample raja]# vi decrement.sh

```
#!bin/bash
echo “ENTER THE INTEGER VALUE: ”
read n
while [ $n –ge 0 ]
do
echo “$n”
let n—
done
```

Result:

Thus the Shell script program to display the decremented integer value was executed successfully.

Ex No: 9 **SEARCH A STRING**

AIM:

To write a shell script to search a string and display it.

Procedure:

1. Create a new vi editor file.

2. Get the name of the destination file, 1st source file and 2nd source file.

3. Concatenate the two files and copied into the destination file. Check the file status with 0.

4. If it is equal then print “File Copied Successfully” otherwise print “Problem copying file”.

PROGRAM: [miet@localhost ~]$ vi co.sh

```
!/bin/bash
string='Haystack';

if [[ $string =~ "Needle" ]]

then

  echo "It's there!"

fi
```

Result:

Thus the shell script to search a string and display it was executed successfully.

EX.NO: 10 **PERFORMING FILE MOMENT USINGCOMMAND LINE ARGUMENTS**

Aim:

To Write a Shell script program that takes three command line arguments. The first argument is the name of the destination file and the other two arguments are names of files to be placed in the destination file.

Procedure:

1. Create a new vi editor file.

2. Get the name of the destination file, 1st source file and 2nd source file.

3. Concatenate the two files and copied into the destination file.Check the file status with 0.

4. If it is equal then print “File Copied Successfully” otherwise print “Problem copying file”.

PROGRAM: [miet@localhost ~]$ vi co.sh

```
!/bin/bash
echo script name : "$0"
echo total number of argument passed: "$#"
echo argument list -
echo 1. $1
echo 2. $2
echo 3. $3
echo all arguments are: "$*"
```

Result:

Thus the program to move the two files into another file using command line arguments was executed successfully.

EX.NO:11 PRINTING THE FILE CONTENT

Aim:

To write a Shell script program to print the content of the file from the

given line number to the next given number of lines.

Procedure:

1. Create a new vi editor file.

2. $# -eq 0, then print the Error msg.

3. If $# -eq 3,

a. If $# -eq 3, then print the content from the given line number.

b. Else print the Error opening file

4. Else print the missing arguments

PROGRAM:

```
if [ $# -eq 3 ]; then
if [ -e $3 ]; then
tail +$1 $3 | head -n $2
else
echo "$0: Error opening file $3"
exit 2
fi
else
echo "Missing arguments!"
fi
```

Result:

Thus the Shell script program to print the content of the file from the given line number to the next given line number was executed successfully.

EX.NO:12 A DISPLAY THE MESSAGE IN LOGIN SESSION

Aim:

To write a Shell script program to say Good morning/Afternoon/Evening

as u log into system.

Procedure:

1. Create a new vi editor file.

2. Set the value of hour.Compare the hour with time of zero, 12and 18.

3. If it is greater than 0 and less than 12 print the string is GOOD MORNING then if it is greater than 12 and less than 18 print the string is GOOD AFTERNOON otherwise print the string is GOOD EVENING

PROGRAM: [miet@localhost ~]$ vi good.sh

```
clear
   hours=`date|cut -c12-13`
   if [ $hours -le 12 ]
   then
        echo "Good Morning"
   else
        if [ $hours -le 16 ]
        then
             echo "Good Afternoon"
        elif [ $hours -le 20 ]
        then
             echo "Good Evening"
else
             echo "Good Night"
        fi
    fi
```

EX.NO:12 B TO PRINT THE DATE IN A DECIDED FORMAT

Aim:

Write a shell script that print out date information in this order: time, day of the week, day number, year– that is like this.21:18:00 IST Thu 4 Feb2016.

Procedure:

1. Create a new vi editor file.

2. Enter the command is date.

3. Print the today date information.

Program [miet@localhost ~]$ vi date.sh

```
#!/bin/bash
now="$(date)"
printf "Current date and time %s\n" "$now"
now="$(date +'%d/%m/%Y')"
printf "Current date in dd/mm/yyyy format %s\n" "$now"
```

Result:

Thus the Shell script program to print the date information was executed successfully.

EX.NO:13 BASIC CALCULATOR USING SWITCH CASE

Aim: To develop a Basic Math calculator Using case Statement.

Procedure:

1) Create a new file.

2) Read the operands.Select any one operation from the list.

3) Perform the operation.Print the result.

Program:[miet@localhost ~]$ vi calculator.sh

```
echo "Enter Two numbers : "
read a
read b
echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch
case $ch in
  1)res=`echo $a + $b | bc`;;
  2)res=`echo $a - $b | bc`;;
  3)res=`echo $a \* $b | bc`;;
  4)res=`echo "scale=2; $a / $b" | bc`;;
esac
echo "Result : $res"
```

Result:

Thus the above program to develop a calculator application was executed successfully.

EX.NO:14 **MULTIPLE CHOICE QUESTION**

Aim:

To Write a Shell script program that presents a multiple choice question,gets the user's answer & report back whether the answer is right, wrong or not in one of the choices .

Procedure:

1. Create a new vi editor file.

2. Enter the command in to file. Enter the options in opt.

3. If the selected option is correct, its shows your choice is right or else wrong.

4. Print the output.

Program [miet@localhost ~]$ vi log.sh

```
#!/bin/bash
Options("option 1",option 2","quit")
Select opt in "$[option [$]]"
Do
Case $ opt in ("option 1")
Echo "your choice is right";;
Case $ opt in ("option 2")
Echo "your choice is wrong"
Quit
break;;
Echo invalid option;;
Esac
Done
```

Result: Thus the above program of multiple choice questions was executed successfully.

EX.NO:15 A **COMMAND LINE ARGUMENT**

Aim:

To Write a Shell script program to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument.

Procedure:

1. Create a new vi editor file.

2. Enter the command in to file.

3. Print the logged on to the system information.

Program [miet@localhost ~]$ vi log.sh

```
#!/bin/bash

if [ $# -ne 1 ]
then
      echo "Usage - $0  file-name"
 exit 1
fi

if [ -f $1 ]
then
   echo "$1 file exist"
else
   echo "Sorry, $1 file does not exist"
fi
```

Output [miet@localhost ~]$ sh log.sh

```
File name: cse

File exist
```

EX.NO:15 B **Report**

Aim: To write a Shell script program that takes a command line argument and reports on whether it's a file or directory, a file or something else.

Procedure:

1. Create a new vi editor file.

2. Get the name of the file or directory.

3. Compare the name of the file or directory.

4. If it is file name print the string is file then if it is directory name print the string is Directory otherwise print the string is Not.

Program

```
#!/bin/bash

echo "Enter a file name:"

read f

if [ -f $f ]

then

echo "File"

elif [ -d $f ]

then

echo "Directory"

else

echo "Not"

fi
```

Result:

Thus the Shell script program to takes a command line argument & reports on whether it's a file or directory or something was executed successfully.