**Experiment 3 Date:** 06/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO2):

Perform system administration task

# Procedure:

1. pwd :- print working directory

$pwd

Output:



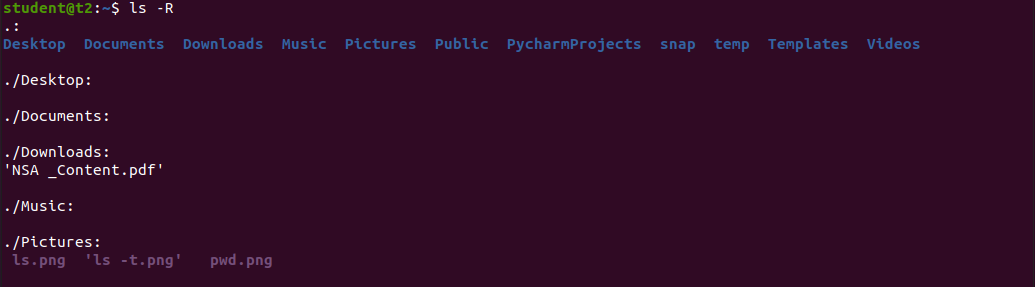
1. ls :- list directory content

$ls

Output:

* 1. ls -R :- list subdirectories recursively

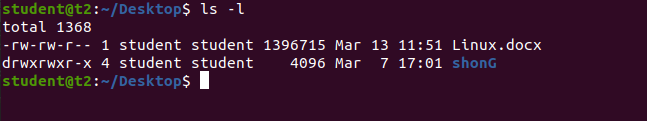
$ls -R

Output:

* 1. ls -l :- Use long listing format

$ls -l

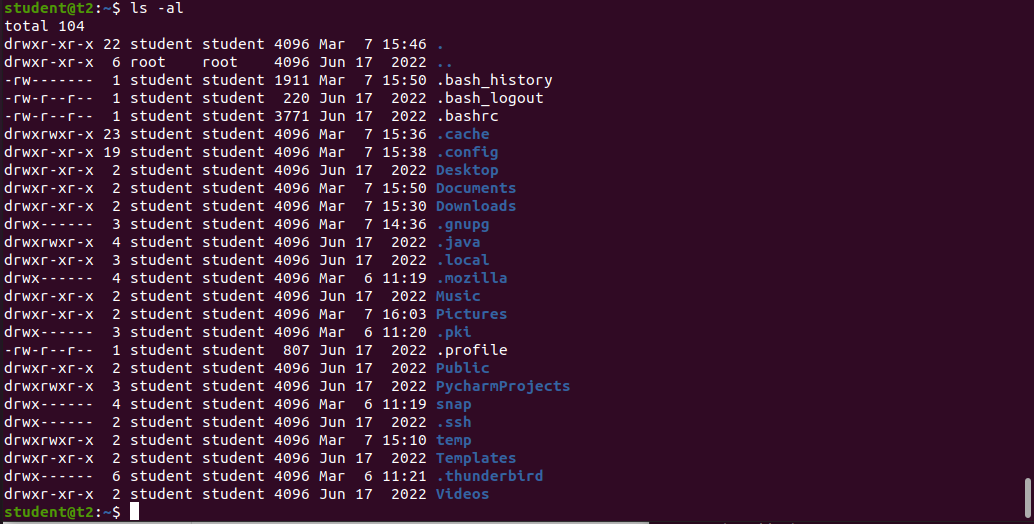
Output:



* 1. ls -al :- list files and directories with detailed information such as permissions, size and owner

$ls -al

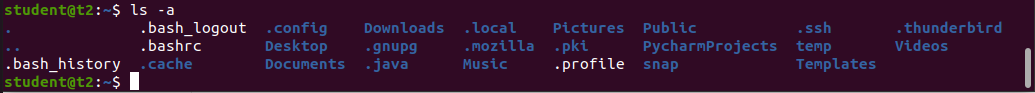
Output:



* 1. ls -a :- List hidden files

$ls -a

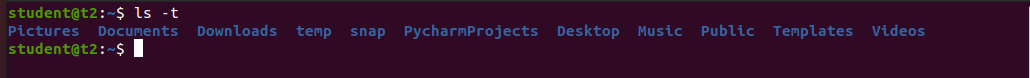
Output:



* 1. ls -t :- Sort by modification time, newest first

$ls -t

Output:



* 1. ls -r :- Reverse order while sorting

$ls -r

Output:

1. history : - Review all previously executed commands right from the shell

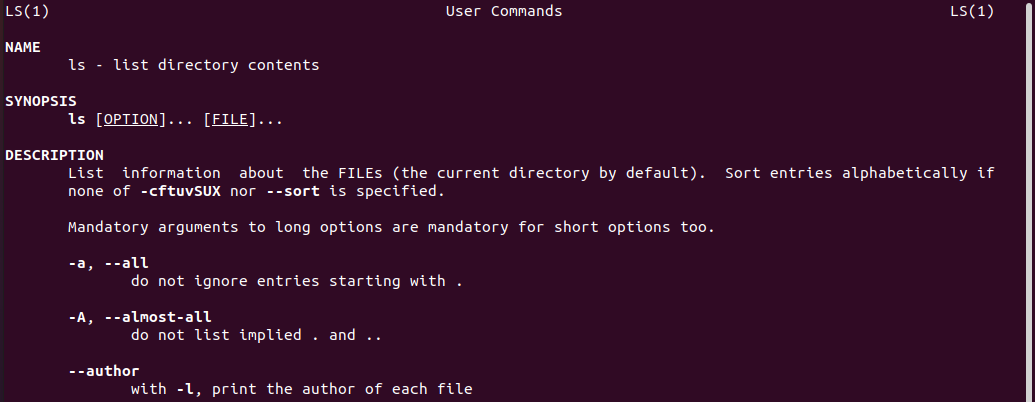
$history

Output:

1. man :- An interface to system reference manuals

$man ls

Output:



1. cd :- Change directory

$cd

Output:



1. cd -- : to go straight to home folder

$cd –

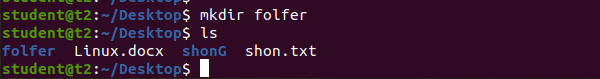
Output :



1. mkdir :- Make directory

$mkdir test

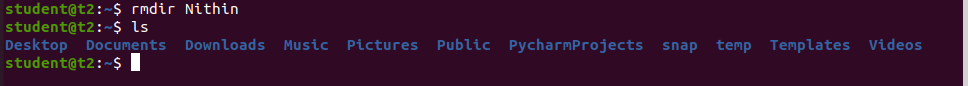
Output:



1. rmdir :- Remove empty directories

$rmdir test

Output:



1. touch :- Create empty file

$touch

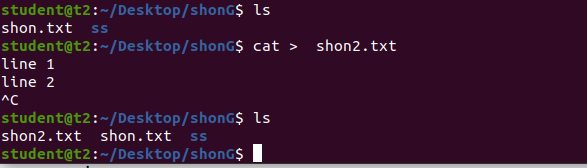
Output:



1. cat :- Concatenate files and print on the standard output
   1. cat > sample.txt :- Create and write in new file

$cat > sample.txt

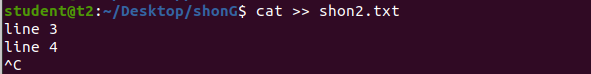
Output:



* 1. cat >> sr.txt :- Append information in already existing file

$cat >> sr.txt

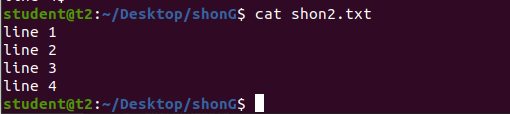
Output:



* 1. cat sample.txt :- Print contents of the file

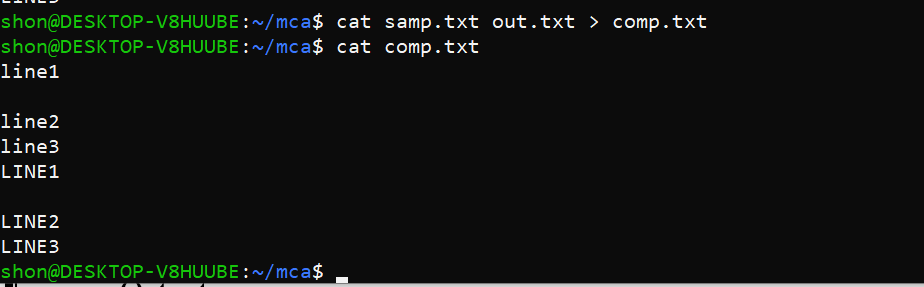
$cat sample2.txt

Output:



* 1. cat sr.txt file.txt > output.txt :- Copy contents of two files to a third new file

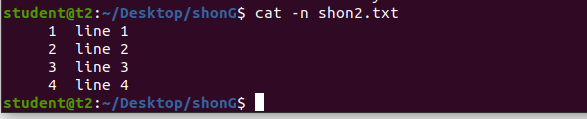
$cat sample sample2 >students.txt

Output: 

* 1. cat -n output.txt :- Number all output lines

$cat -n students.txt

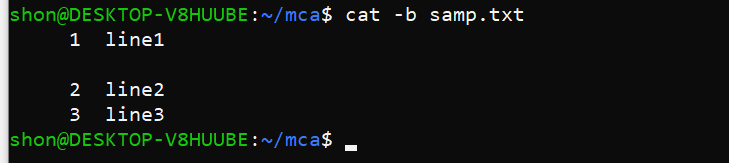
Output:



* 1. cat -b sr.txt :- Remove numbering for empty lines

$cat -b sr.txt

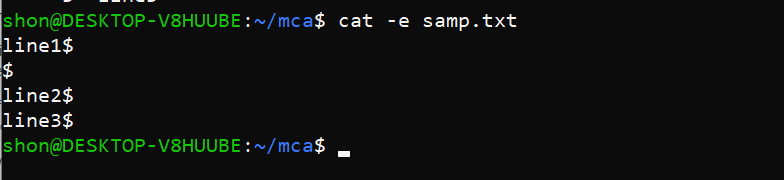
Output:



* 1. cat -e students :- Display $ at end of each line

$cat -e students.

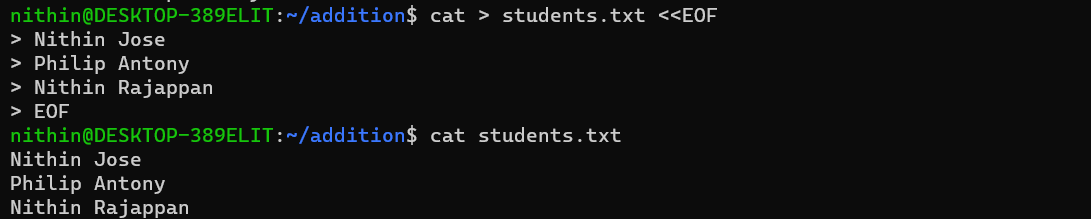
Output:



* 1. cat << EOF :- Displays an end marker at the end of a file.

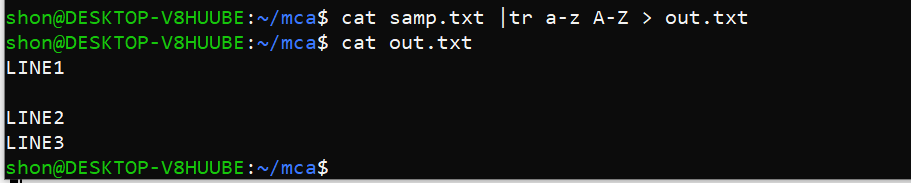
$cat > students.txt <<EOF

Output:



* 1. cat file.txt | tr a-z A-Z > output1.txt :- To change content to uppercase

$cat file1.txt | tr a-z A-Z > output.txt

Output: 

# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 4: Date:** 07/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO2):

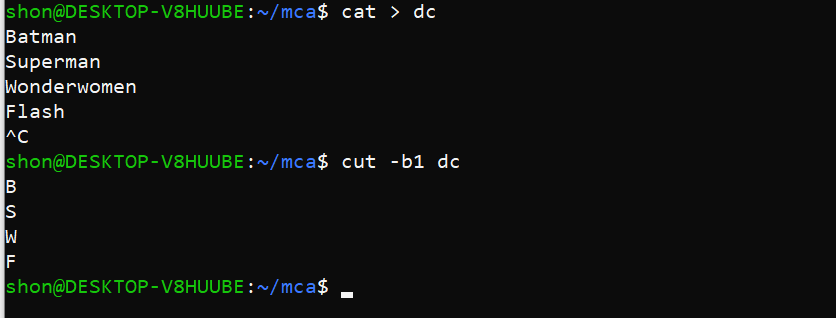
Perform system administration task

# Procedure:

1. cut :- For cutting out the sections from each line of files and writing the result to standard output
   1. cut -b1 :- Cut by first byte position

$cut -b1 file1.txt

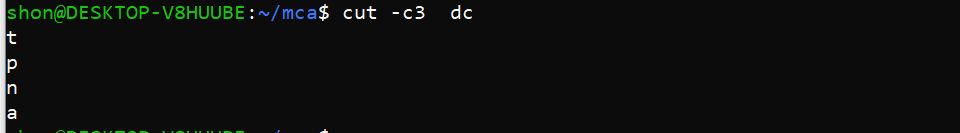
Output:



* 1. cut -c3 :- Cut by third character

$cut -c3 dc\_comics.txt

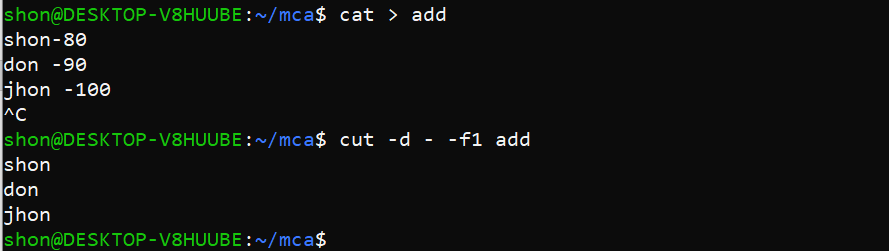
Output:



* 1. cut -d - -f1 file.txt :- Cut by delimiter

$cut -d - -f1 sample.txt

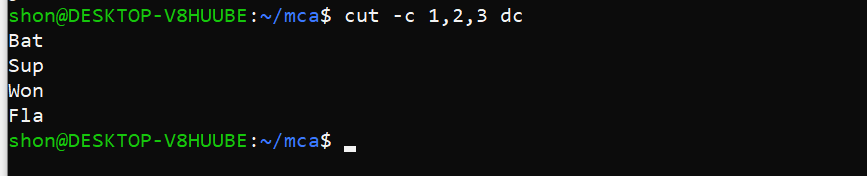
Output:



* 1. cut -c :- Select only these characters

$cut -c 1,3,5 sample.txt

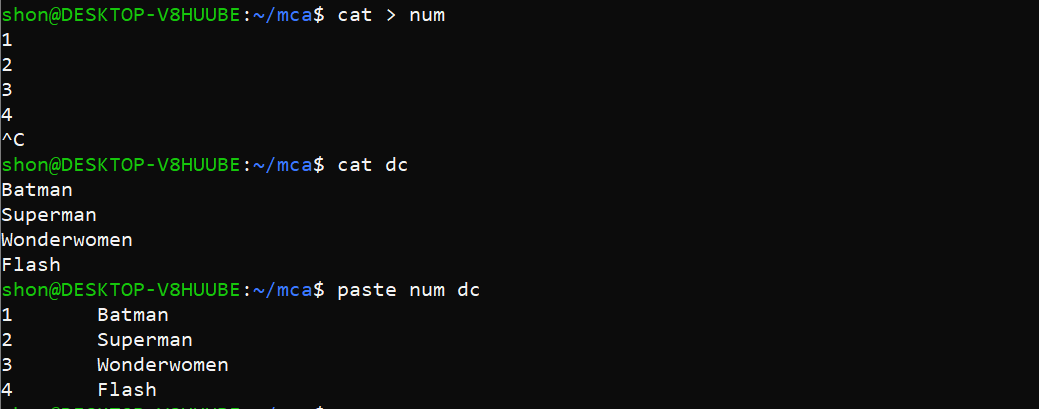
Output:



1. Paste :- Merge lines of files

$paste sr.txt file1.txt

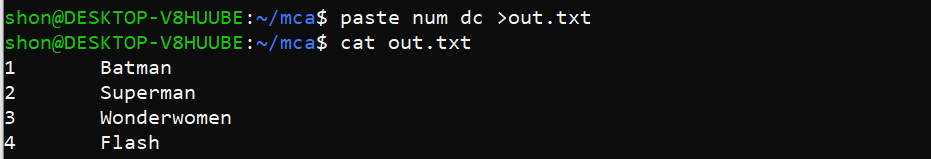
Output:



* 1. paste file1.txt output.txt> output3.txt :- Paste the merged content to new file

$paste file1 file2 > file3

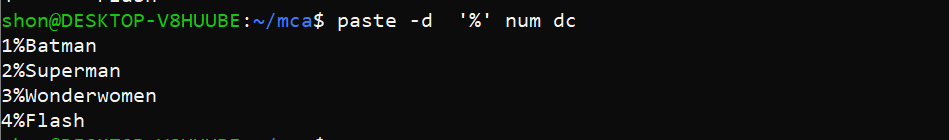
Output:



* 1. paste -d ‘%’ file3.txt output.txt :- Separate the merged parts using a symbol(%)

$paste -d ‘%’ file1 file2

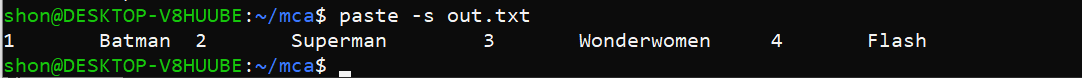
Output:



* 1. paste -s output.txt :- Display output in a single line

$paste -s file3

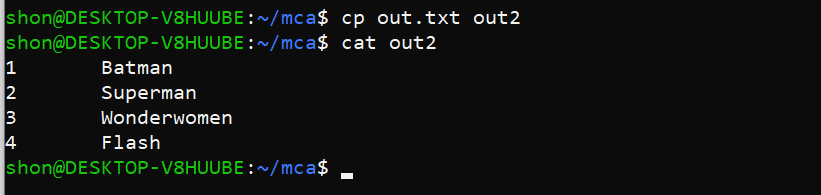
Output:



1. cp :- Copy the content
   1. cp file3.txt sr.txt : -Overwrite existing file

$cp file1 file2

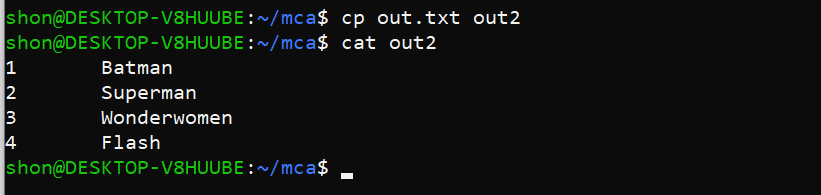
Output:



* 1. cp sr.txt output5.txt :- Copy into new file

$cp file3 file4

Output:



* 1. cp -r class mca :- Copy directories and subdirectories from existing directory to a new one

$cp -r addition mca

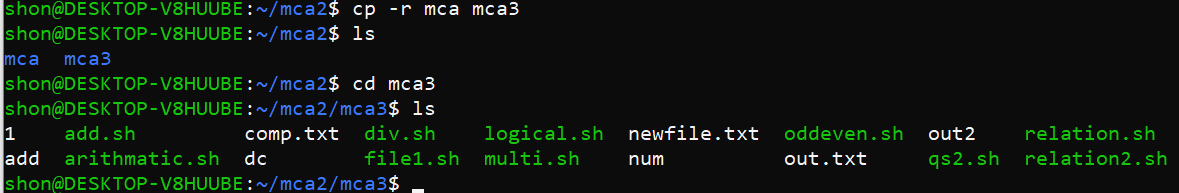
Output:



* 1. cp newfile class:- Copy file from one directory to another

$cp -r mca class

Output:



# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 5 Date**: 13/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO2):

Perform system administration task

# Procedure:

1. read :- Read content of one line of input into a variable

$read

echo $REPLY :- To print the input from the default variable

Output:

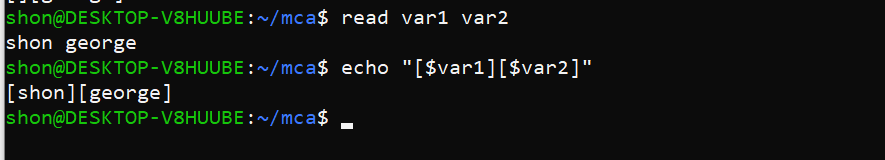


* 1. read var1 var2 var3 :- To read into specific variables

$read var1 var2 var3

$echo “[$var1][$var2][$var3]”

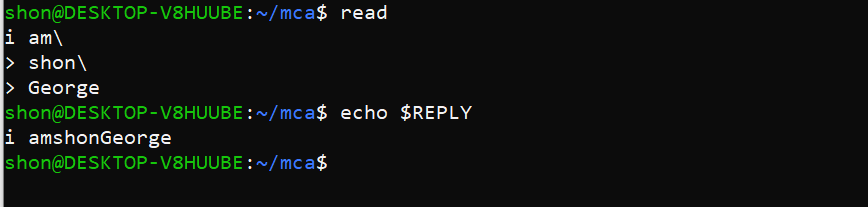
Output:



* 1. read input \ :- To read multiple lines

$read My \ name is \ Shon

Output:

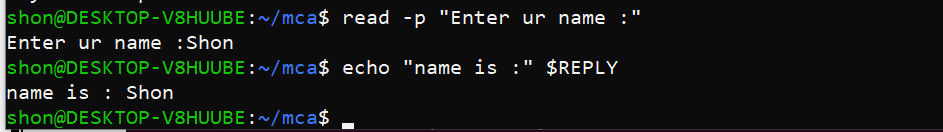


* 1. read -p :- Prompt text from user

$read -p “Enter your name”

$echo “my name is $REPLY”

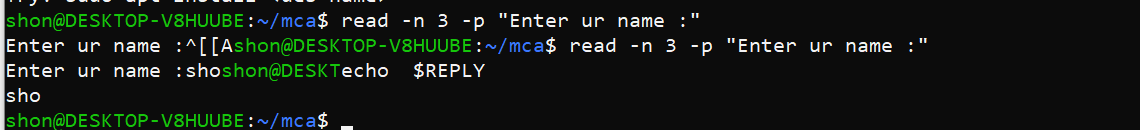
Output:



* 1. read -n :- Specify limit

$read -n 8 -p “Enetr your name”

Output:



* 1. read -s :-For security. Hides input

$read -s -p “Enter the password: “

Output:



1. wc :- Word count display number of lines, number of words, number of bytes and file name

$wc profile

Output:



* 1. wc -l :- Display number of lines
  2. wc -m :-Display number of bytes
  3. wc -c :- Display number of characters
  4. wc -w :-Display number of words

$ wc -l profile

$ wc -m profile

$ wc -c profile

$ wc -w profile

Output:



* 1. wc -L :- Displays length of longest line

$ wc -L profile

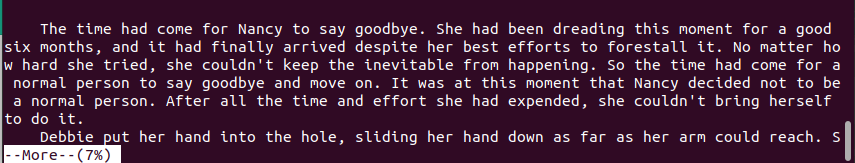
Output:



1. more :- It is similar to cat to display the content. The difference is that in case of larger files, cat command output will scroll off your screen while more command display output one screenful at a time.

$ more longtext.txt

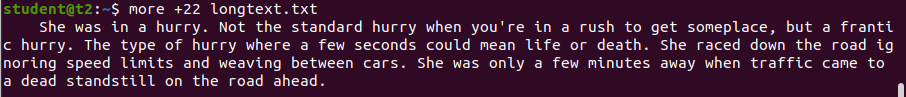
Output:



* 1. more +15 file.txt :- Will display content after the specified number of lines

$ more +22 lontext.txt

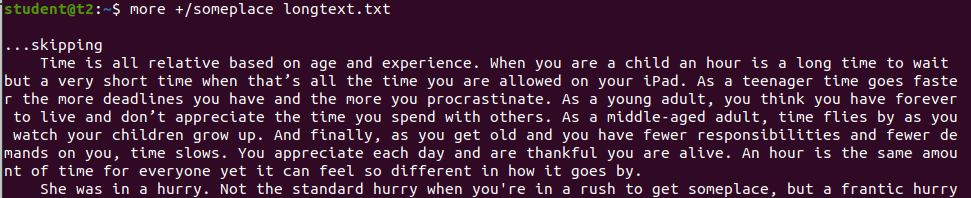
Output:



* 1. more +/pattern file.txt :- Search and navigate towards a particular string and view all the instances.

$ more +/someplace longtext.txt

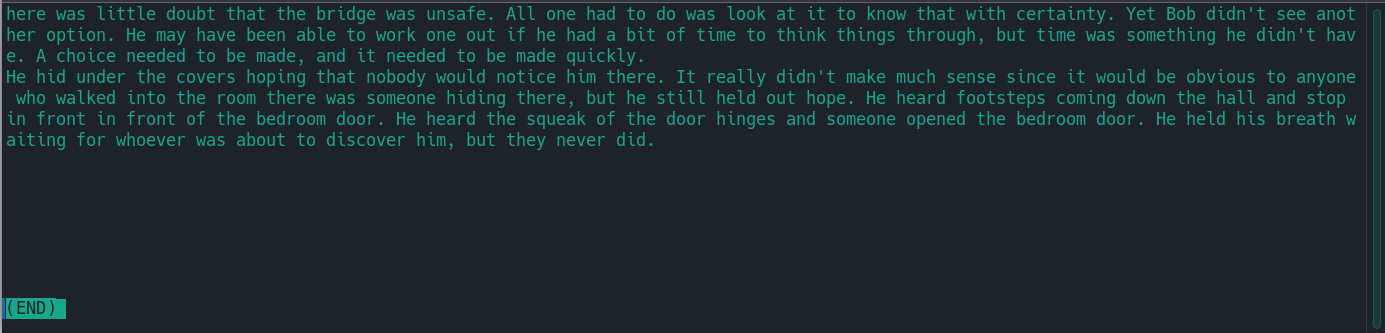
Output:



* 1. more -p file.txt :- Clear the whole screen and then display the text.

$ more -p longtext.txt

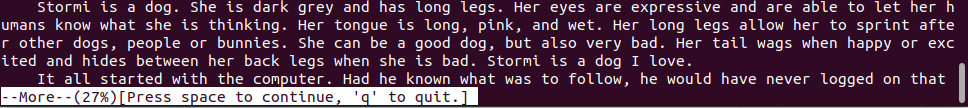
Output:



* 1. more -d file.txt :- Helps the user to navigate according to instructions, [space to continue and ‘q’ to quit]

$ more -d longtext.txt

Output:



# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 6: Date:** 14/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO2):

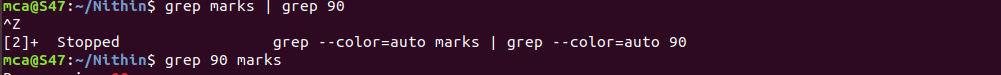
Perform system administration task

# Procedure:

1. grep :- Filtering and Searching content easily

$ grep 90 marks

Output:



* 1. grep -i :- Case insensitive search of a particular content￼

$ grep -i English marks

Output:



* 1. grep -v :- Exclude that content during search￼￼￼￼

$ grep -v English marks

Output:



* 1. grep -A1 :- Specific content and one line after the content

$ grep -A1 English marks

Output:



* 1. grep -B1 :- Specific content and one line before the content

$ grep -B1 English marks

Output:



* 1. grep -C1 :- Specific content and one lone before and after the content.

$ grep -C1 English marks

Output:



1. head :- Used to display the first content of the file(Top 10 lines by default)

$ head demo

Output:-



* 1. head -number filename:- Specific number of lines

$ head -5 demo

Output:



1. tail:- Used to display last contents of a file( last 10 by default)

$ tail demo

Output:



* 1. tail -number filename :- Specific number of content from last

$ tail -5 demo

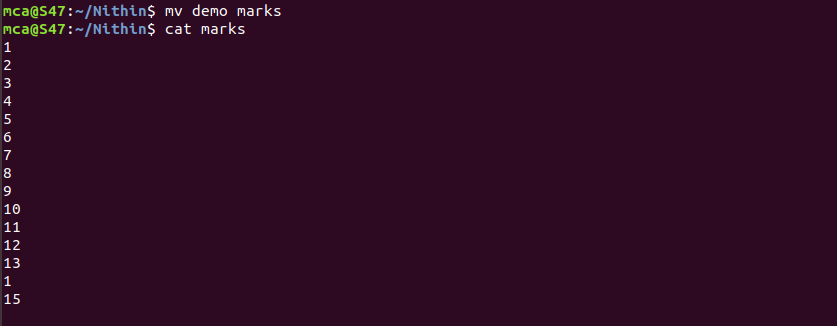
Output:



1. mv :- move from one location to another or it can be used to rename a file. Content will be overwritten.

$ mv demo marks

Output:-



* 1. mv -b :- To take backup of a file while moving.

$ mv -b marks profile

Output:



* 1. mv -i :- Prompt confirmation from user before overwriting.

$ mv -i profile profile1

Output:



# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 7 Date:** 20/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO2):

Perform system administration task

# Procedure:

1. expr :- Evaluate the given expression and display the output.

$expr 12 + 8

Output:



* 1. expr x + y :- Add two variables obtained through read

$read x

$read y

$expr $x + $y

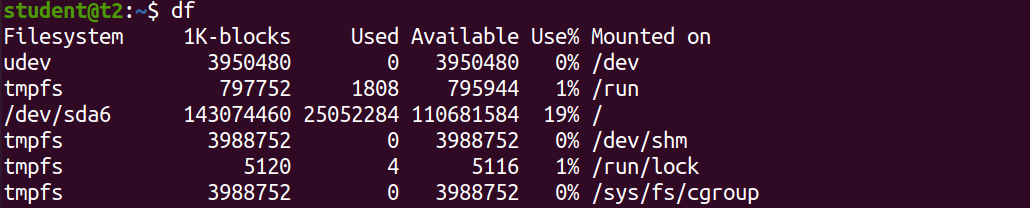
Output:



1. df :- Get a report on disk utilization of the system

$df

Output:



1. du :- check how much space a file or directory in a given directory

$du dc\_comics

Output:



1. sudo :- superuser do
   1. sudo useradd user :- Add new user

$sudo useradd nithin

Output:



* 1. sudo passwd user :- Update password of the user

$sudo passwd nithin

Output:



* 1. sudo groupadd -g identifier name:- To create new group

$sudo groupadd -g 765 mcastd



* 1. sudo usermod -G name user :- Add users to group

$sudo usermod -G mcastd nithin



* 1. id user :- Details on group name and numeric id of particular user.

$id nithin

Output:



compgen -g :- Display all the groups created

$compgen -g

Output:



1. chmod :- Used to change the access permissions of files and directories. It stands for change mod namely, read(r), write(w), execute(x)
   1. chmod -wx file :- deny permission to write and execute for file

$chmod -wx nithin

Output:



* 1. chmod +wrx file :- give permission to write, read and execute for a file

$chmod +wrx file

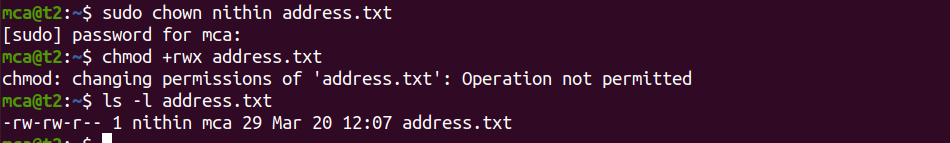
Output:



1. sudo chown :- Used to change ownership of a file or directory for a user or a group. It stands for change owner.

$sudo chown nithin address.txt

Output:



1. sudo userdel user :- Delete user

$sudo userdel nithin

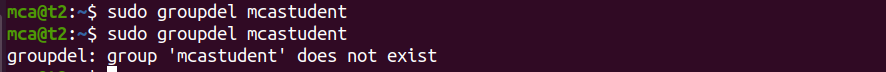
Output:



1. sudo groupdel name :- Delete group

$sudo groupdel mcastudent

Output



# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 8 Date:** 21/03/2023

# Aim:

Familiarization of Linux Commands

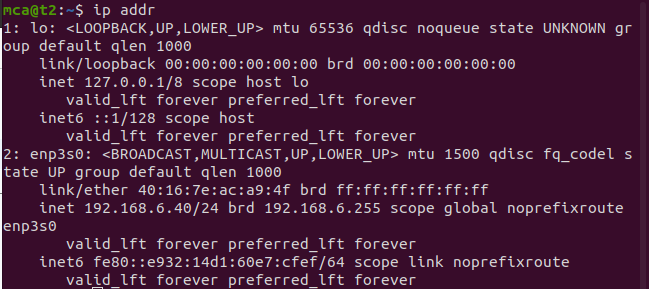
# Course Outcome(CO2):

Perform system administration task

# Procedure:

1. ip addr:- Get ip address of the system

$ip addr

Output:

1. ssh user@ip address:- Stands for Secure Shell Protocol used to securely connect to a remote server or system. ssh is secure in the sense that it transfers data in encrypted form between host and client.

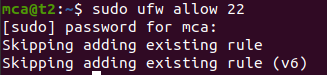
$ssh [mca@192.168.6.39](mailto:mca@192.168.6.39)

Output:

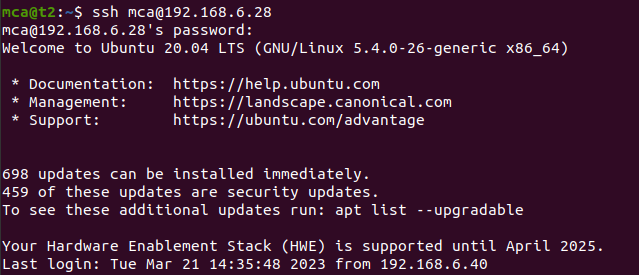
* 1. sudo apt-get install openssh -server :- Update port
  2. sudo ufw allow 22

$sudo ufw allow 22

Output:

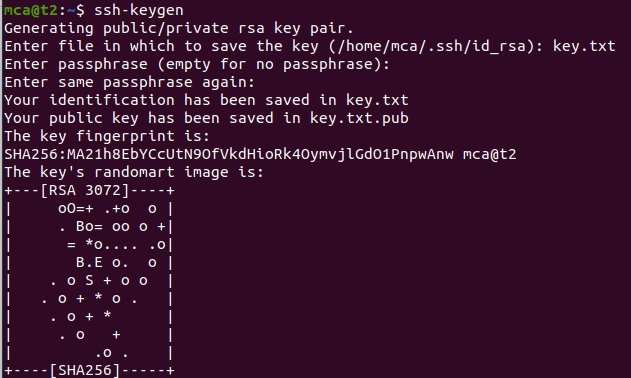


* 1. $ssh [mca@192.168.6.28](mailto:mca@192.168.6.28)

Output:

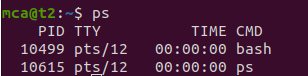
* 1. ssh-keygen :- Generating a key for secure shell

$ssh-keygen

Output:

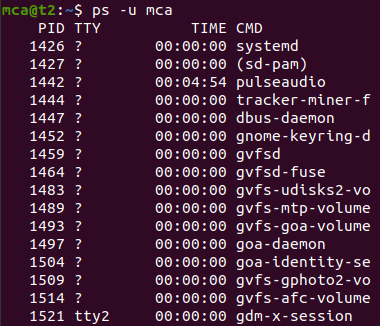
1. ps :- Stands for Process. Currently running programs and running instances.
   1. $ps

Output:



* 1. ps -u :- Display all running processes of a particular user

$ps -u mca

Output:

* 1. ps -C :- Specific process

$ps -C firefox

Output:

* 1. ps -f –p PID :- List the process by id

$ps -f -p 2762

Output:

# Result:

Output displayed successfully and CO2 was obtained.

**Experiment 9 Date:** 28/03/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO4):

Write shell scripts required for system administration

# Procedure:

1. Shell script to display date:

$vi filename.sh :- Open Editor by creating a shell script file.

Press ‘i’ to INSERT

#!/bin/bash :- To indicate the shell used date

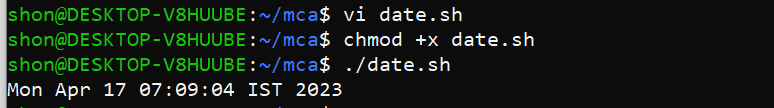
Press ‘Esc’ to end INSERT

:wq! :- To save and quit

chmod +x filename.sh :- To give execution permission

./filename.sh :- To execute shell script

Output:



1. Shell script to display your name:

$vi filename.sh Press ‘i’ to INSERT #!/bin/bash

echo “What is your name?” read name

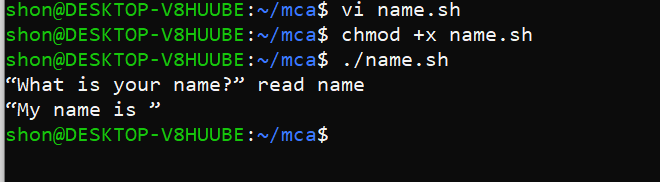
echo “My name is $name” Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Multiple Commands (ls, pwd, date, mkdir) in Shell Script:

$vi filename.sh Press ‘i’ to INSERT #!/bin/bash

date

ls

pwd

mkdir

file6

ls

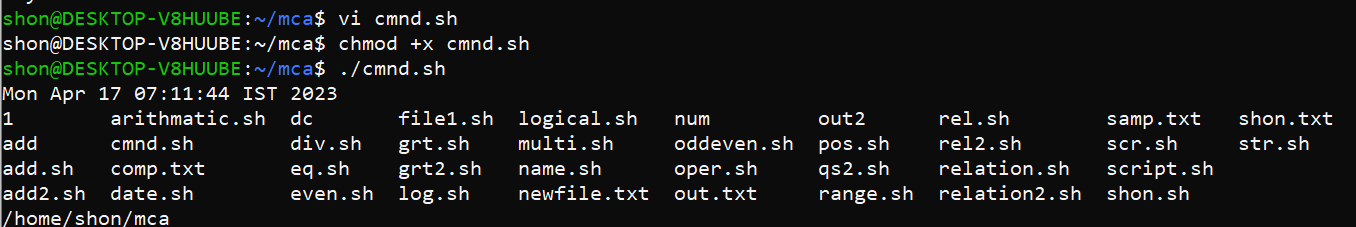
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to demonstrate variables

$vi filename.sh Press ‘i’ to INSERT #!/bin/bash

echo “Enter your name: “ read name

echo “Your name is $name”

echo “File Name: $0”

echo “First Parameter: $1”

echo “Second Parameter: $2”

echo “Quoted Values: $@”

echo “Quoted Values: $\*”

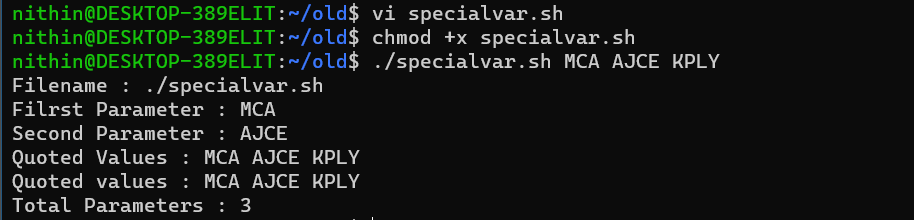
echo “Total Number of Parameters: $#” Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to count lines and words in a file

readlink -f filename :- Get path of required file

$vi filename.sh Press ‘i’ to INSERT #!/bin/bash

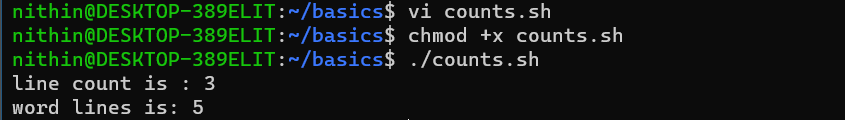
file\_path = “/home/Reqfilename.sh” countlines = `wc –lines < $file\_path` countwords = `wc –words < $file\_path` echo “Number of lines: $countlines” echo “Number of words: $countwords” Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to display array index

$vi filename.sh Press ‘i’ to INSERT #!/bin/bash

Name=("name1" "name2" "name3" "name4")

echo "First Index: ${Name[0]}"

echo "Second Index: ${Name[1]}"

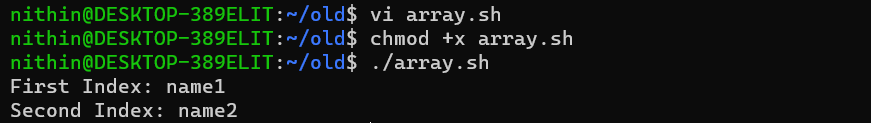
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



# Result:

Output displayed successfully and CO4 was obtained.

**Experiment 10 Date:** 03/04/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO4):

Write shell scripts required for system administration

# Procedure:

1. Shell script to add two number: vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

val=`expr 10 + 10`

echo "total is $val"

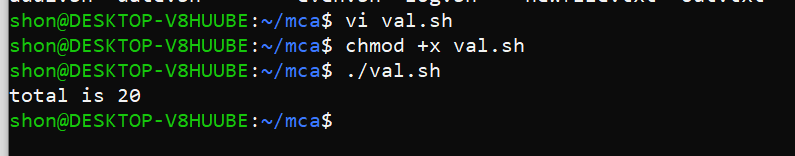
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Write a shell script to initialize two numeric variables. Then perform addition operation on both values and store the result in the third variable.

vi filename.sh Press ‘i’ to INSERT #!/bin/bash

read -p "Enter first number " num1

read -p "Enter second number " num2

sum=$(($num1 + $num2))

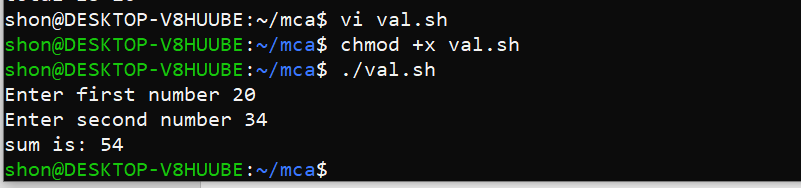
echo "sum is: $sum"Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to read two numbers as command line parameters and perform the addition operation

vi filename.sh Press ‘i’ to #!/bin/bash

a=10

b=20

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

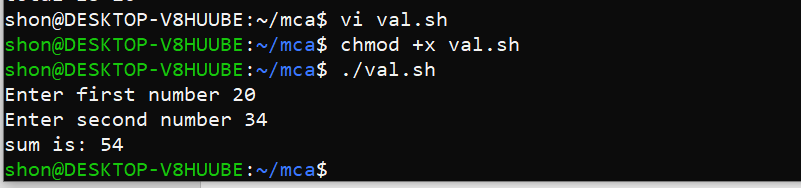
echo "sum is: $sum"Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh num1 num2

Output:



1. Shell script which takes input from the user at run time and then calculate the sum of given number and store to a variable and show the result vi filename.sh Press ‘i’ to INSERT#!/bin/bash sum=$(($1 + $2)) echo "sum is $sum"

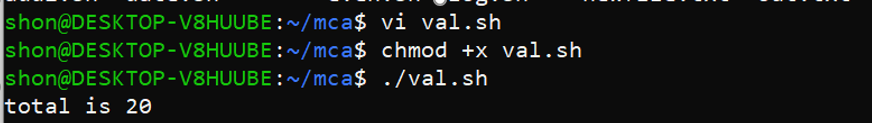
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh num1 num2

Output:



1. Shell script to demonstrate Arithmetic operators (addition, subtraction, multiplication, division, modulus, increment, decrement) by taking user input and store to another variable

vi filename.sh Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the First number: " num1 read -p "Enter the Second number: " num2 sum=$(( $num1 + $num2 ))

prd=$(( $num1 \* $num2 )) diff=$(( $num1 - $num2 )) quo=$(( $num1 / $num2 )) rem=$(( $num1 % $num2 )) echo "Sum : $sum"

echo "Product : $prd" echo "Difference : $diff" echo "Quotient : $quo" echo "Remainder : $rem" if [ $num1 == $num2 ] then

echo "$num1 is equal to $num2"

fi

if [ $num1 != $num2 ] then

echo "$num1 is not equal to $num2"

fi

(( ++num1 ))

echo "Increment operator on first number: $num1" (( --num2 ))

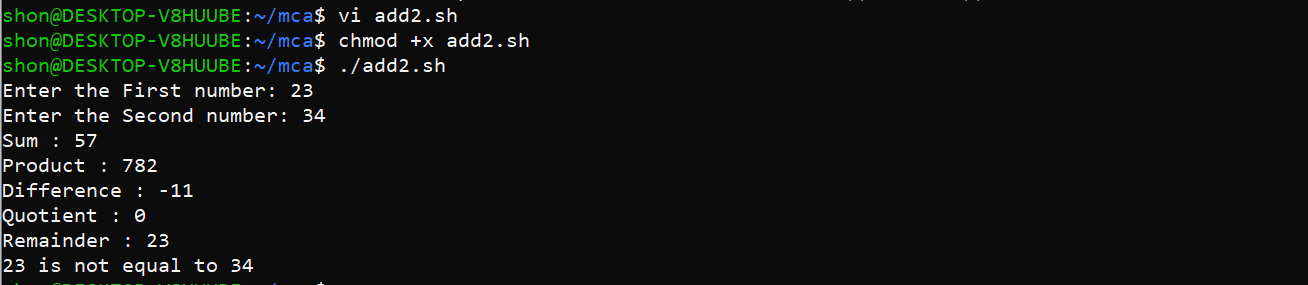
echo "Decrement operator on second number: $num2" Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



# Result:

Output displayed successfully and CO4 was obtained.

**Experiment 11 Date:** 04/04/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO4):

Write shell scripts required for system administration

# Procedure:

1. Shell script to demonstrate Relational operators (equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to) by taking user input

vi filename.sh Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter First : " i

read -p "Enter Second : " j

if (( $i == $j ))

then

echo "Both Numbers are equal"

else

echo "Both Numbers are different"

fi

if (( $i != $j ))

then

echo "Both Numbers are different"

else

echo "Both Numbers are Equal"

fi

if (( $i < $j ))

then

echo "First is less than Second Number"

else

echo "First is not less than Second Number"

fi

if (( $i <= $j ))

then

echo "First is less than or equal to Second Number"

else

echo "First is not less than or equal to Second Number"

fi

if (( $i > $j ))

then

echo "First is greater than Second Number"

else

echo "First is not greater than Second Number"

fi

if (( $i >= $j ))

then

echo "First is greater than or equal to Second Number"

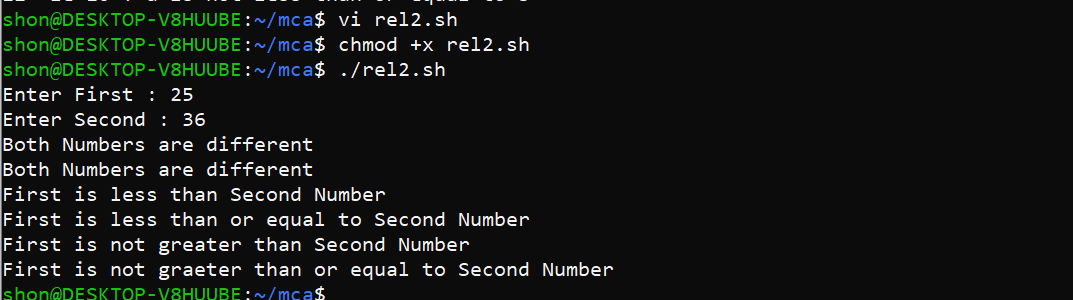
else

echo "First is not graeter than or equal to Second Number"

fi

echo "Invalid Inputs"

Output:



1. Shell script to demonstrate Relational operators (equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to)

vi filename.sh Press ‘i’ to INSERT

#!/bin/bash

a=12

b=10

if [ $a -eq $b ]

then

echo "$a -eq $b : a is equal to b"

else

echo "$a -eq $b : a is not equal to b"

fi

if [ $a -ne $b ]

then

echo "$a -ne $b : a is not equal to b"

else

echo "$a -ne $b : a is equal to b"

fi

if [ $a -gt $b ]

then

echo "$a -gt $b : a is greater than to b"

else

echo "$a -gt $b : a is not graeter than b"

fi

if [ $a -lt $b ]

then

echo "$a -lt $b : a is less than b"

else

echo "$a -lt $b : a is not less than b"

fi

if [ $a -ge $b ]

then

echo "$a -ge $b : a is greater than or equal to b"

else

echo "$a -ge $b : a is not greater than or equal to b"

fi

if [ $a -le $b ]

then

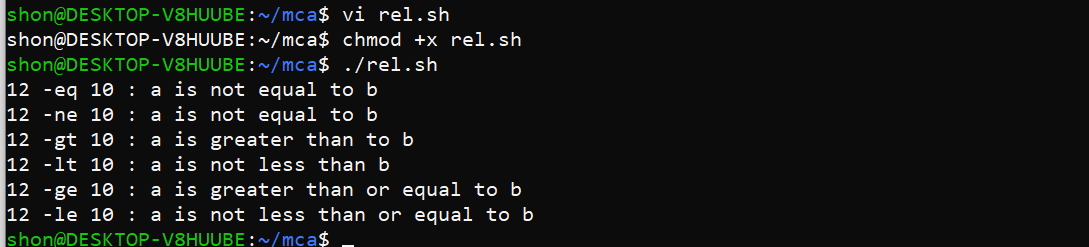
echo "$a -le $b : a is less than or equal to b"

else

echo "$a -le $b : a is not less than or equal to b"

fi

Output:



1. Shell script to demonstrate Logical operators (AND, OR, NOT) by taking user input

#!/bin/bash

read -p "Enter a : " a

read -p "Enter b : " b

if (( $a == "true" & $b == "true" ))

then

echo "Both are true"

else

echo "Both are not true"

fi

if (( $a == "true" || $b == "true" ))

then

echo "Atleast one of the input is true"

else

echo "None of the input are true"

fi

if(( ! $a == "true" ))

then

echo "a was initially false"

else

echo "a was initially true"

fi

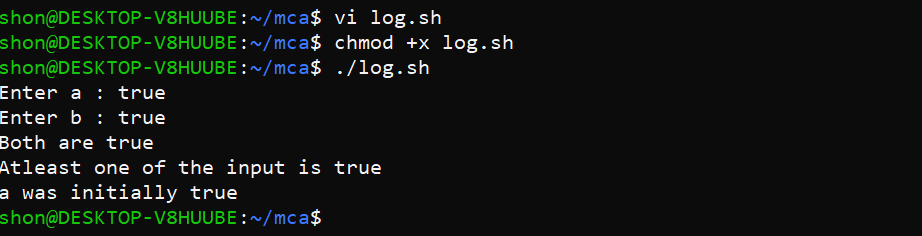
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Write a shell script to check if a number is even or odd. vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the Number to check : " num

if(( $num == 0 ))

then

echo "The Entered Number is Zero"

else

if(( num % 2 == 0 ))

then

echo "$num is even"

else

echo "$num is odd"

fi

fi

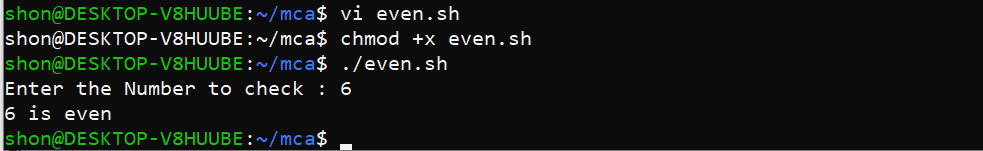
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Write a shell script to check whether a number is positive or negative vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the Number to check : " num

if(( $num > 0 ))

then

echo "The Entered Number is Positive"

elif(( $num < 0 ))

then

echo "The entered Number is Negative"

elif(( $num ==0 ))

then

echo "The Eneterd Number is Zero"

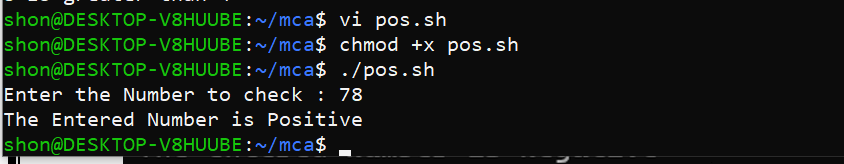
else

echo "Please enter a valid Number"

fi

./filename.sh

Output:



1. Write a shell script to find the greatest of two numbers vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the Number 1 : " num1

read -p "Enter the Number 2 : " num2

if(( $num1 > $num2 ))

then

echo "$num1 is greater than $num2"

elif(( $num2 > $num1 ))

then

echo "$num2 is greater than $num1"

else

echo "$num1 and $num2 are equal"

fi

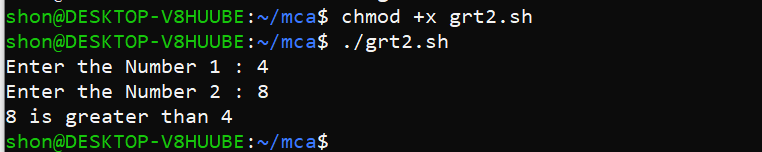
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Write a shell script to find the greatest of three numbers vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter Number 1 : " num1

read -p "Enter Number 2 : " num2

read -p "Enter Number 3 : " num3

if(( $num1 > $num2 ))

then

if(( $num1 > $num3 ))

then

echo "$num1 is the greatest of three numbers entered"

else(( $num3 > $num1 ))

echo "$num3 is the greatest of three numbers entered"

fi

elif(( $num2 > $num1 ))

then

if(( $num2 > $num3 ))

then

echo "$num2 is the greatest of three numbers entered"

else(( $num3 < $num2 ))

echo "$num3 is the greatest of three numbers entered"

fi

else

if(( $num1 > $num3 ))

then

echo "Both Number1 and Number2 are the greatest Numbers"

else(( $num1 < $num3 ))

echo "$num3 is the greatest among three numbers"

fi

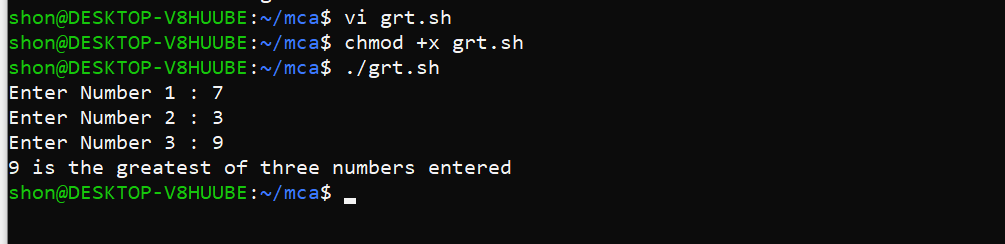
fi

:wq!

chmod +x filename.sh

./filename.sh

Output:



# Result:

Output displayed successfully and CO4 was obtained.

**Experiment 12 Date:** 11/04/2023

# Aim:

Familiarization of Linux Commands

# Course Outcome(CO4):

Write shell scripts required for system administration

# Procedure:

1. Shell script to demonstrate String operators (Equal, Not Equals, Size is zero, Size is non-zero, Empty string) by taking user input

vi filename.sh Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the string 1: " a

read -p "Enter the string 2: " b

if [ $a == $b ]

then

echo "Both strings are the same"

else

echo "Both strings are not the same"

fi

if [ $a != $b ]

then

echo "Both Strings are not the same"

else

echo "Both strings are the same"

fi

if [ -z $a ]

then

echo "-z $a :Entered string is null"

else

echo "-z $a :Entered string was not null string"

fi

if [ -n $a ]

then

echo "-n $a : Entered string is not null"

else

echo "-n $a : Entered string is null"

fi

if [ $a ]

then

echo " $a : Entered string is not null"

else

echo " $a : Entered string is null"

fi

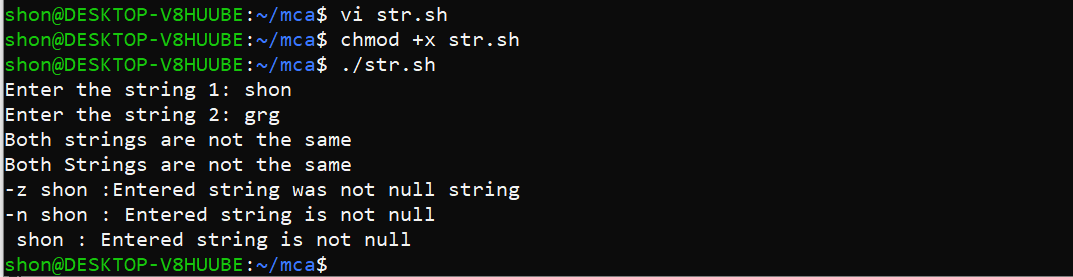
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to demonstrate Bitwise operators (AND, OR, XOR, Complement, Right Shift, Left Shift) by taking user input

vi filename.sh Press ‘i’

#!/bin/bash

read -p "Enter first no: " a

read -p "Enter second no: " b

bitwiseAND=$(( a&b ))

echo "BITWISE AND : $bitwiseAND"

bitwiseOR=$(( a|b ))

echo "BITWISE OR : $bitwiseOR"

bitwiseXOR=$(( a^b ))

echo "BITWISE XOR : $bitwiseXOR"

complement=$(( ~a ))

echo "BITWISE COMPLEMENT : $complement"

leftshift=$(( a<<1 ))

echo "LEFT SHIFT : $leftshift"

rightshift=$(( b>>1 ))

echo "RIGHT SHIFT : $rightshift"

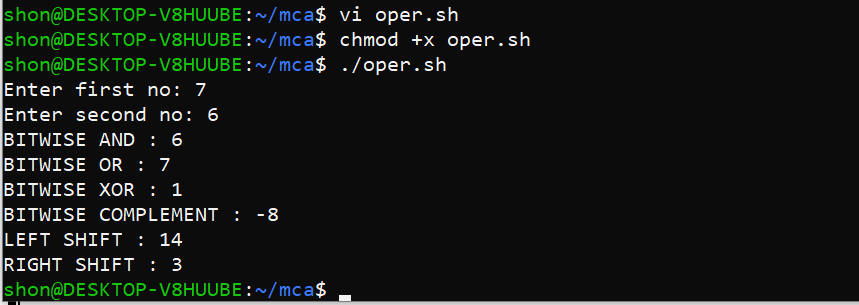
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



3. Shell script to demonstrate File Test operators (Exist(e), Size(s), Read Permission(r),  
Execute Permission(x), Write Permission(w)) by taking user input  
vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter file name: " f1

if [ -e $f1 ]

then

echo "$f1 exist"

else

echo "$f1 does not exist"

fi

if [ -s $f1 ]

then

echo "$f1 is not empty"

else

echo "$f1 is empty"

fi

if [ -r $f1 ]

then

echo "$f1 has read permission"

else

echo "$f1 does not have read permission"

fi

if [ -x $f1 ]

then

echo "$f1 has execute permission"

else

echo "$f1 does not have execute permission"

fi

if [ -w $f1 ]

then

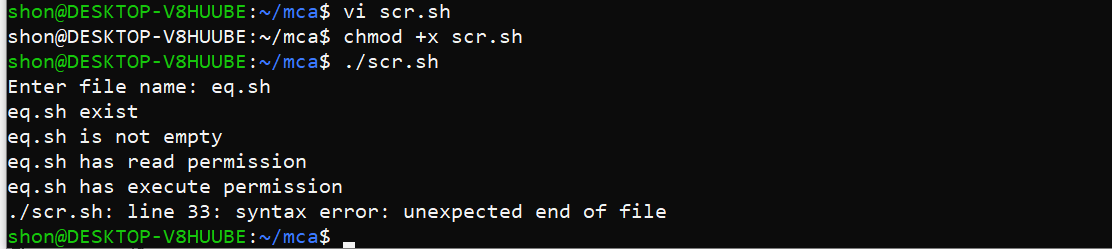
echo "$f1 has write permission"

else

echo "$f1 does not have write permission"

fi

Output:



4. Shell Script to check if two numbers are equal using if statement  
vi filename.sh  
Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the first number: " num1

read -p "Enter the second number: " num2

if (( $num1 == $num2 ))

then

echo "Both numbers are equal"

fi

if (( $num1 != $num2 ))

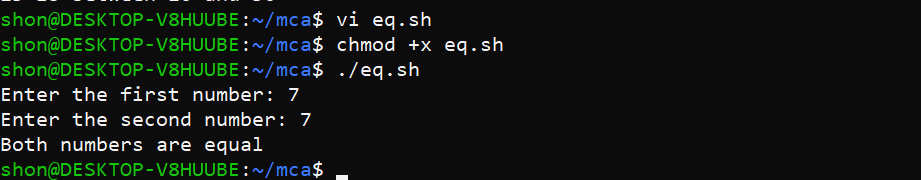
then

echo "Both numbers are not equal"

fi

Press ‘Esc’ to end INSERT

:wq!  
chmod +x filename.sh  
./filename.sh  
Output:



5. Shell Script to check the range of a number if numbers using else if ladder  
vi filename.sh

Press ‘i’ to INSERT  
 #!/bin/bash

read -p "Enter the number(b/w 0-50): " num1

if (( $num1 >= 0 && $num1 <= 10 ))

then

echo "$num1 is between 0 and 10"

elif (( $num1 >= 11 && $num1 <= 20 ))

then

echo "$num1 is between 10 and 20"

elif (( $num1 >= 21 && $num1 <= 30 ))

then

echo "$num1 is between 20 and 30"

elif (( $num1 >= 31 && $num1 <= 40 ))

then

echo "$num1 is between 30 and 40"

elif (( $num1 >= 41 && $num1 <= 50 ))

then

echo "$num1 is between 40 and 50"

fi  
Press ‘Esc’ to end INSERT  
:wq!  
chmod +x filename.sh  
./filename.sh  
Output:

