**Experiment 3**

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

Perform system administration task

**Procedure:**

1. pwd :- print working directory

$pwd

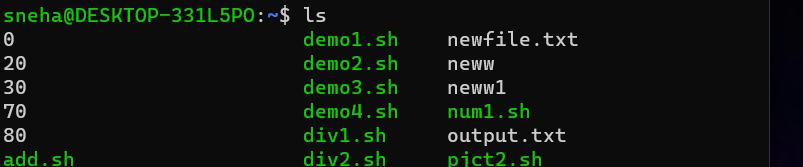
Output:



1. ls :- list directory content

$ls

Output:



1. ls -R :- list subdirectories recursively

$ls -R

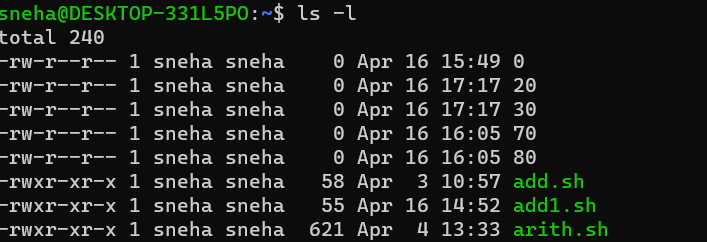
Output:



1. ls -l :- Use long listing format

$ls -l

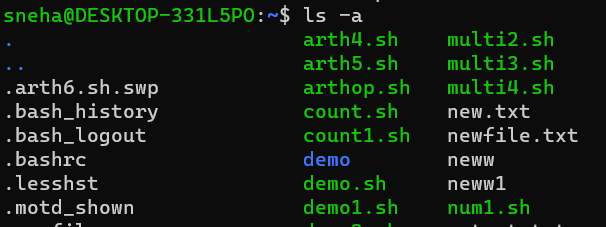
Output:



1. ls -a :- List hidden files

$ls -a

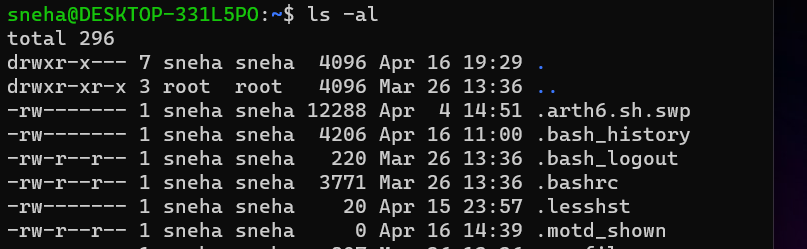
Output:



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

$ls -al

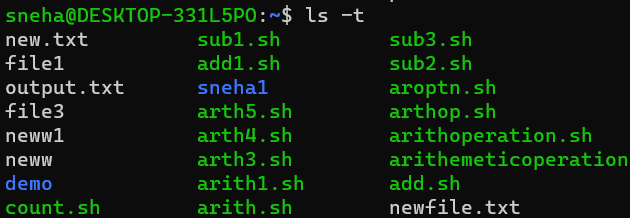
Output:



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

$ls -t

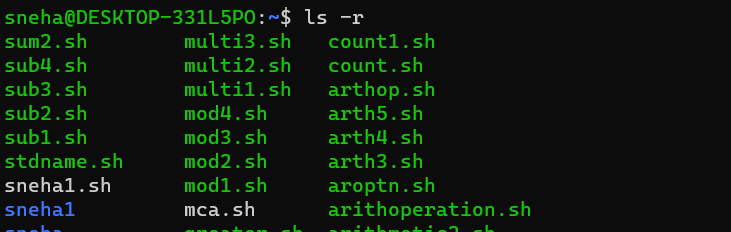
Output:



1. ls -r :- Reverse order while sorting

$ls -r

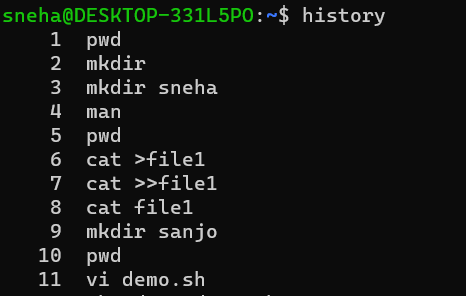
Output:



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

$history

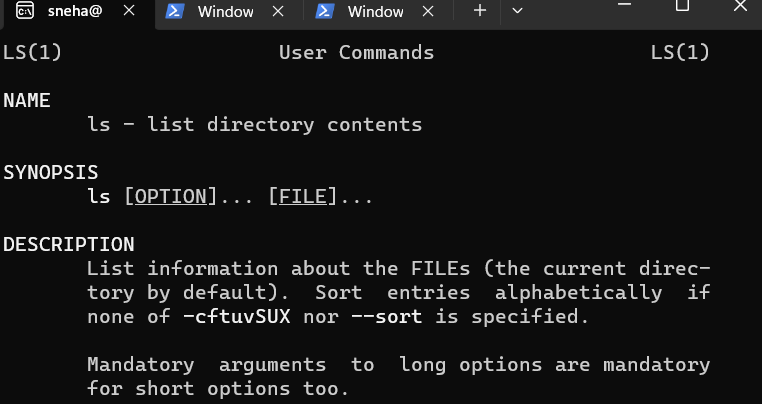
Output:



4. man :- An interface to system reference manuals

$man ls

Output:



5. cd :- Change directory

$cd

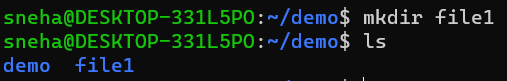
Output:



6. mkdir :- Make directory

$mkdir demo

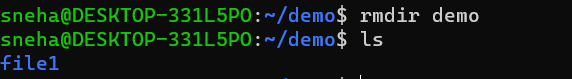
Output:



7. rmdir :- Remove empty directories

$rmdir demo

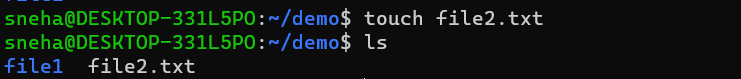
Output:



8. touch :- Create empty file

$touch

Output:

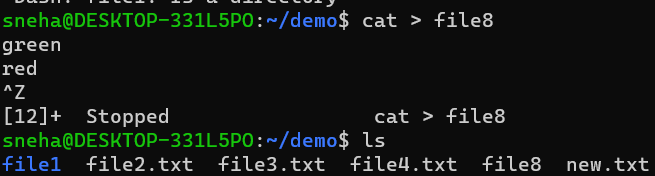


9. cat :- Concatenate files and print on the standard output

1. cat > file8.txt :- Create and write in new file

$cat > file8.txt

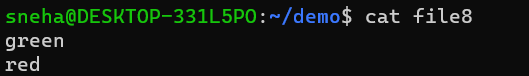
Output:



1. cat file8.txt:- Print contents of the file

$cat file8.txt

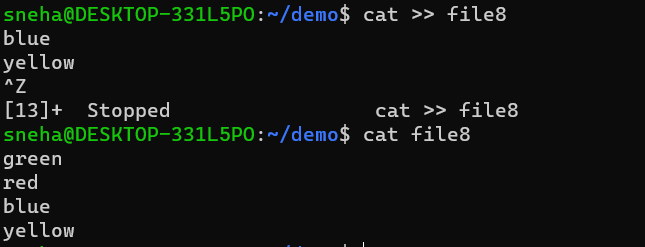
Output:



1. cat >> file8.txt:- Append information in already existing file

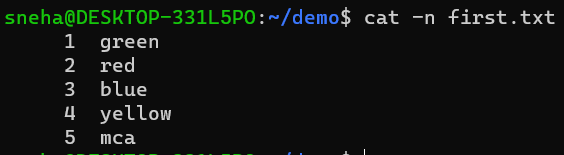
$cat >> file8.txt

Output:



1. cat file8.txt file9.txt >first.txt :- Copy contents of two files to a third new file

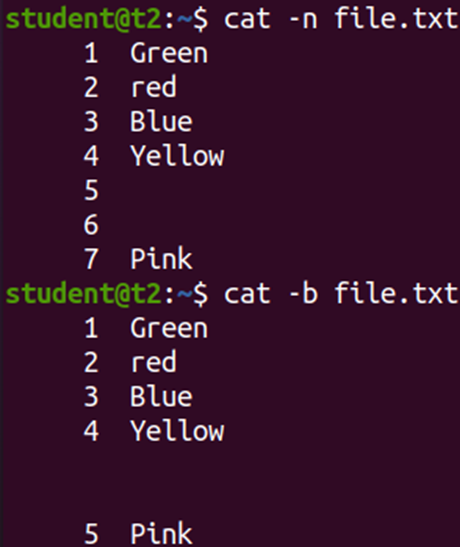
$cat first.txt



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

$cat -b sr.txt

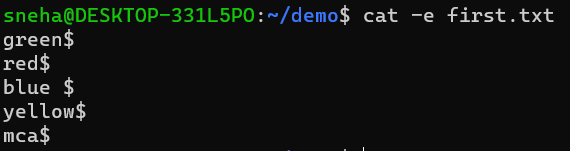
Output:



f)cat -e output.txt :- Display $ at end of each line

$cat -e output.txt

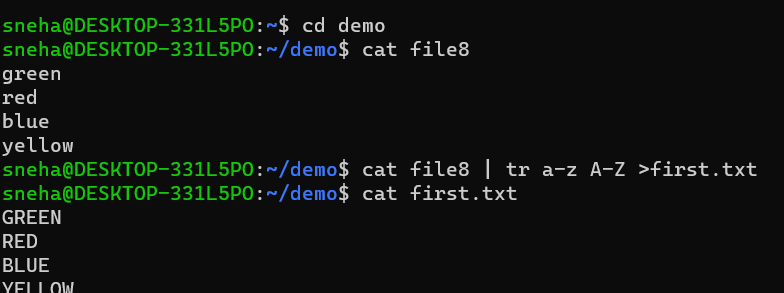
Output:



h)cat file1.txt | tr a-z A-Z > first.txt :- To change content to uppercase

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

Output:



**Result:**

Output displayed successfully and CO2 was obtained.

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

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

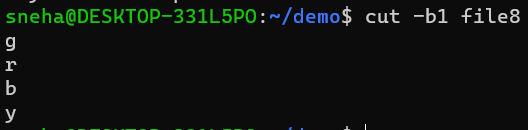
Perform system administration task

**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

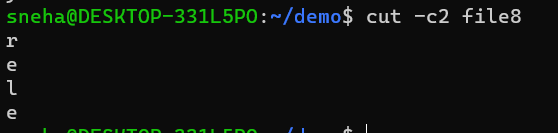
Output:



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

$cut -c3 file1.txt

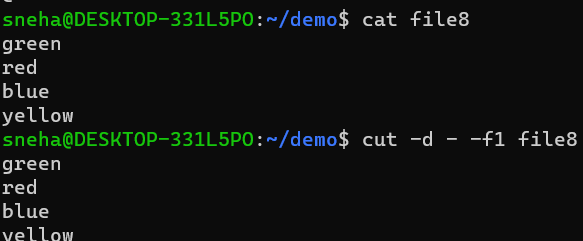
Output:



* 1. cut -d - -f1 file3.txt :- Cut by delimiter

$cut -d - -f1 file3.txt

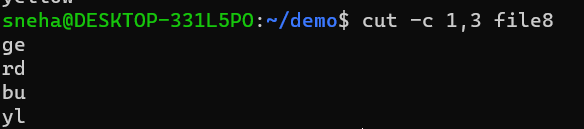
Output:



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

$cut -c 1,3,5 file3.txt

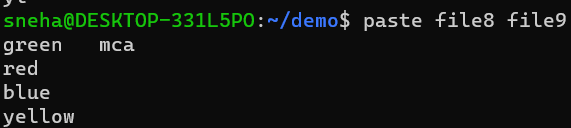
Output:



1. Paste :- Merge lines of files

$paste sr.txt file1.txt

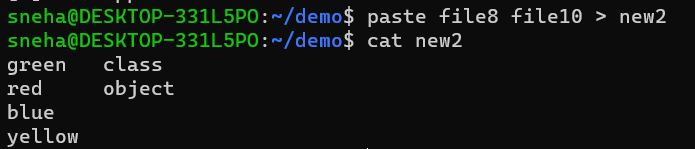
Output:

cat

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

$paste file1.txt output.txt > output3.txt

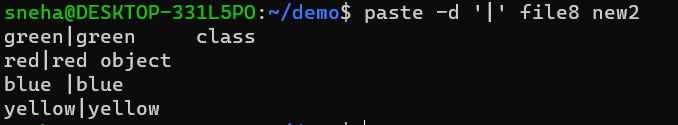
Output:



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

$paste -d ‘%’ file3.txt output.txt

Output:



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

$paste -s output.txt

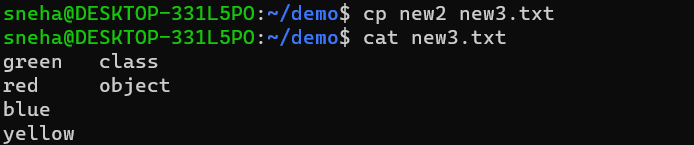
Output:



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

$cp file3.txt sr.txt

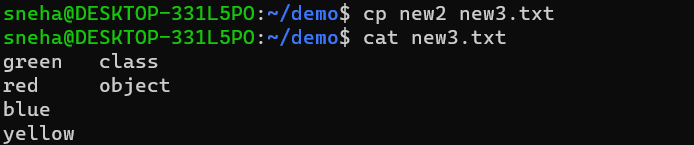
Output:



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

$cp sr.txt output5.txt

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

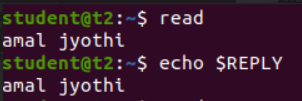
**Procedure:**

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

$read

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

Output:



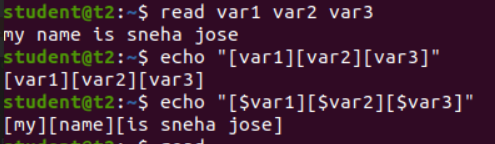
echo $REPLY :- To print the input from the default variable

* 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

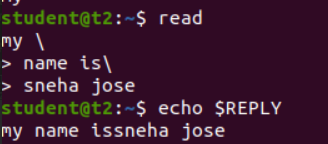
$read

My \

name is \

Sreerag

Output:

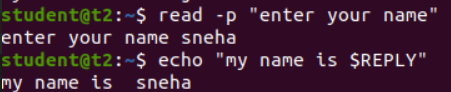


* 1. read -p :- Prompt text from user

$read -p “Enter your name”

$echo “my name is $REPLY”

Output:



* 1. read -n :- Specify limit

$read -n 6 -p “Enter six characters only”

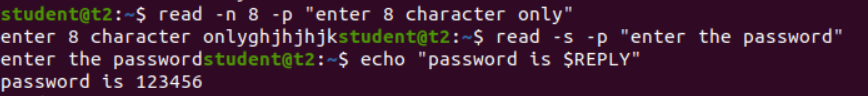
Output:



* 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 file.txt

Output:



* 1. wc -l :- Display number of lines

$wc -l file.txt

Output:



* 1. wc -m :-Display number of bytes

$wc -m file.txt

Output:



* 1. wc -c :- Display number of characters

$wc -c file.txt

Output:



* 1. wc -w :-Display number of words

$wc -w file.txt

Output:



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

$wc -L file.txt

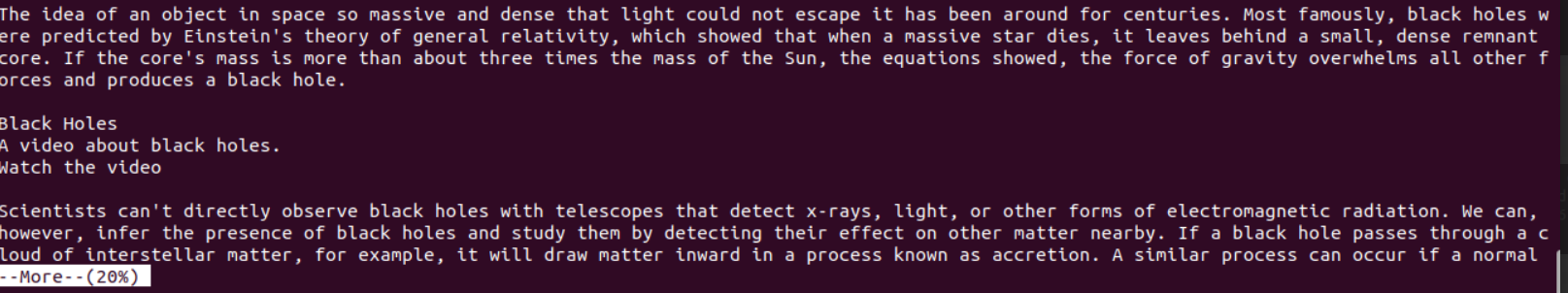
Output:



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 finalfantasy.txt

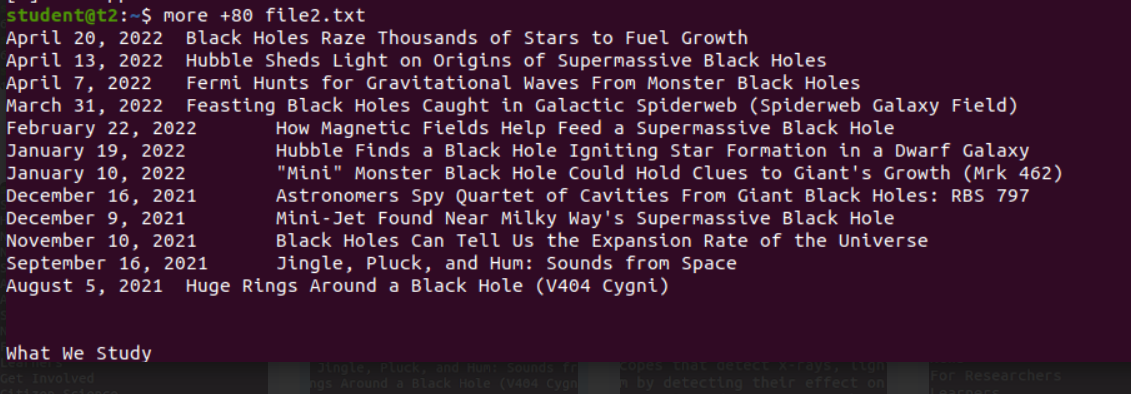
Output:



* 1. more +80 file.txt :- Will display content after the specified number of lines

$more +15 finalfantasy.txt

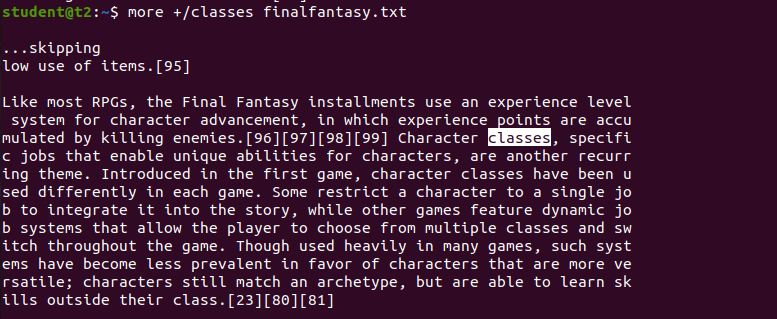
Output:



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

$more +/classes finalfantasy.txt

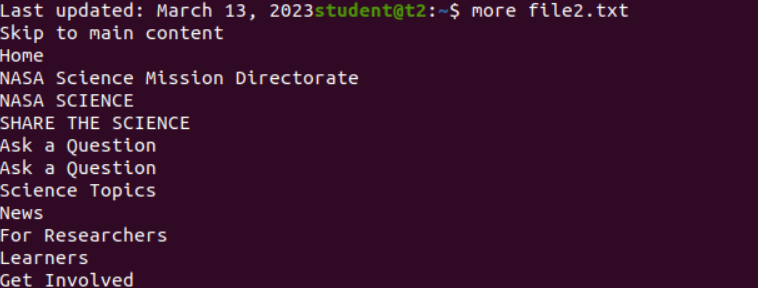
Output:



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

$more -p file.txt

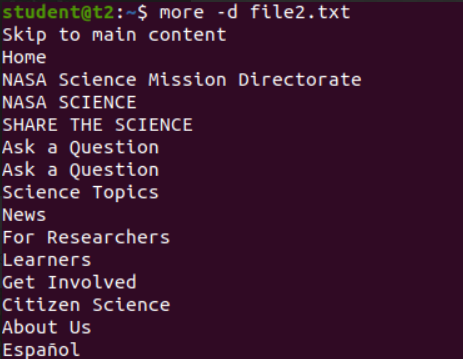
Output:



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

$more -d finalfantasy.txt

Output:



**Result:**

Output displayed successfully and CO2 was obtained.

**Experiment 6:**

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

Perform system administration task

**Procedure:**

1. grep :- Filtering and Searching content easily

$gerp 34 Mark

Output:



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

$grep -i 42 Mark

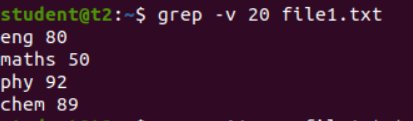
Output:



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

$grep -v 42 Mark

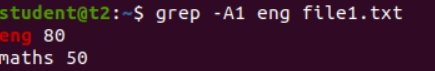
Output:



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

$grep -A1 Maths Mark

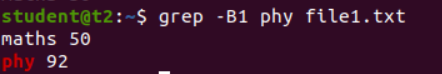
Output:



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

$grep -B1 Maths Mark

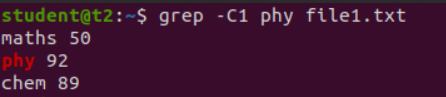
Output:



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

$gerp -C1 Maths Mark

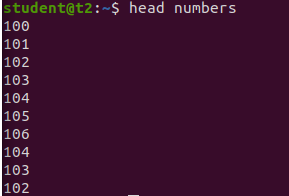
Output:



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

$head numbers

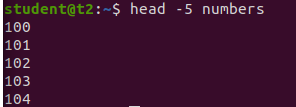
Output:-



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

$head -5 numbers

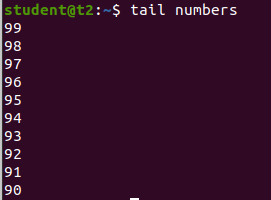
Output:



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

$tail numbers

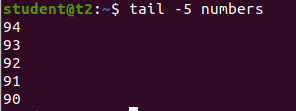
Output:



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

￼￼$tail -5 numbers

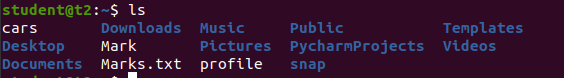
Output:



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

$mv numbers Mark

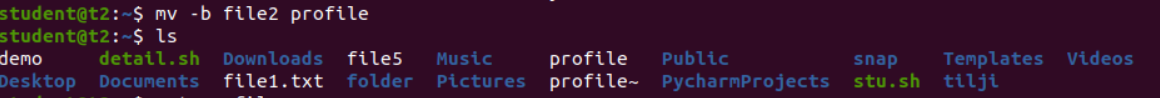
Output:-



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

$mv -b Mark profile

Output:



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

$mv -i Marks.txt profile

Output:



**Result:**

Output displayed successfully and CO2 was obtained.

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

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

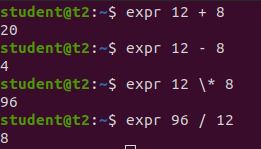
Perform system administration task

**Procedure:**

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

$expr 12 + 8

Output:



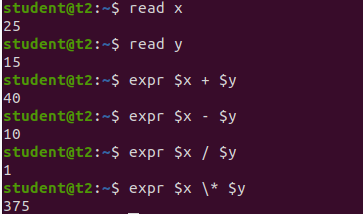
* 1. expr x + y :- Add two variables obtained through read

$read x

$read y

$expr $x + $y

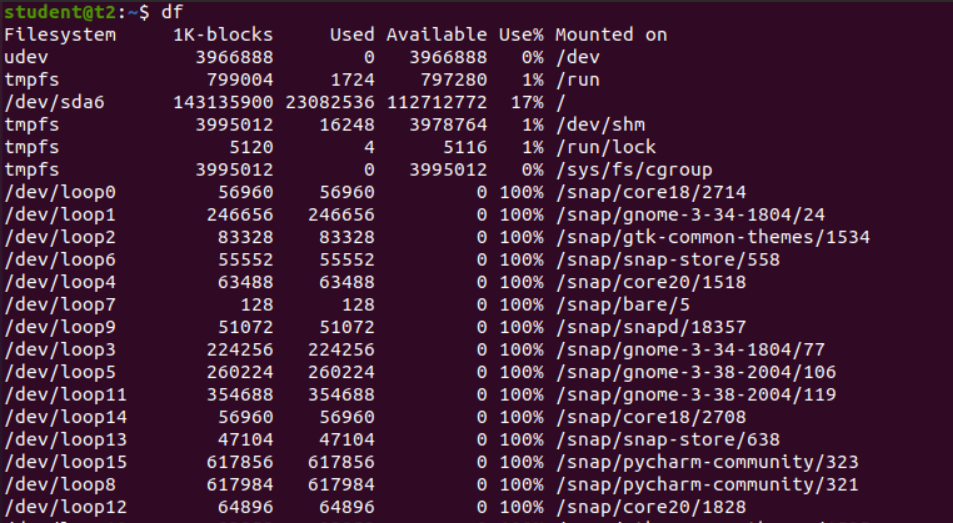
Output:



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 file.txt

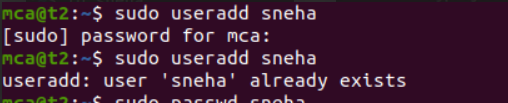
Output:



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

$sudo useradd Sreerag

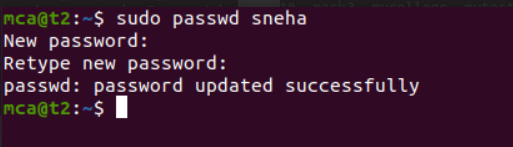
Output:



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

$sudo passwd Sreerag

Output:



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

$sudo groupadd -g 269 mcastd

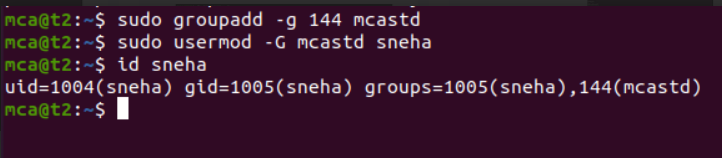
* 1. sudo usermod -G name user :- Add users to group

$sudo usermod -G mcastd Sreerag

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

$id Sreerag

Output:



1. 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 file

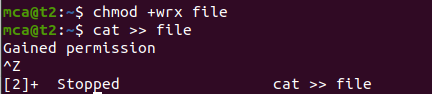
Output:



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

$chmod +wrx file

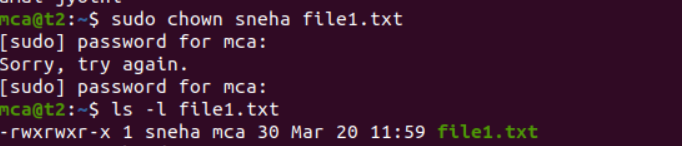
Output:



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 Sreerag file

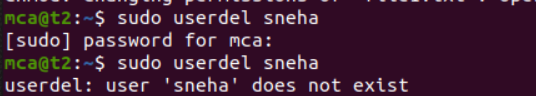
Output:



1. sudo userdel user :- Delete user

$sudo userdel Sreerag

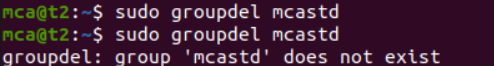
Output:



1. sudo groupdel name :- Delete group

$sudo groupdel mcastd

Output



**Result:**

Output displayed successfully and CO2 was obtained.

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

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

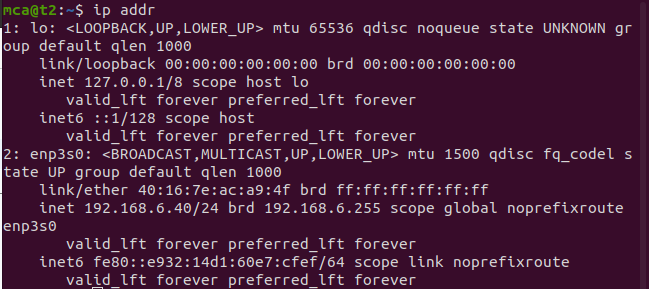
Perform system administration task

**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

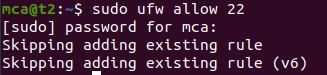
Output:



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

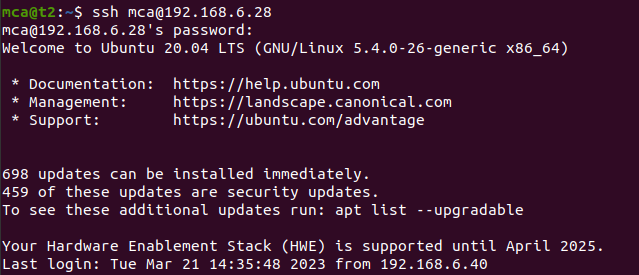
$sudo ufw allow 22

Output:



* 1. $ssh mca@192.168.6.28

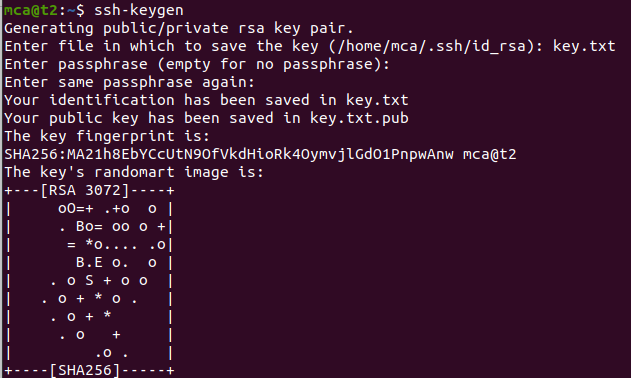
Output:



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

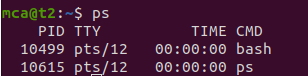
$ssh-keygen

Output:



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

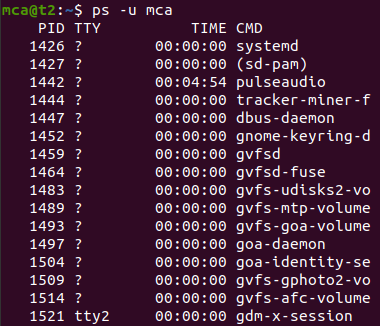
Output:



* 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

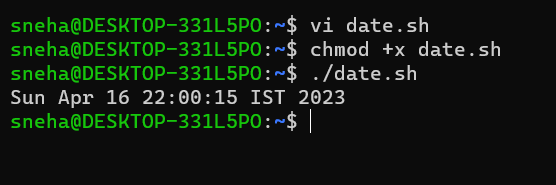
Press ‘Esc’ to end INSERT

: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”

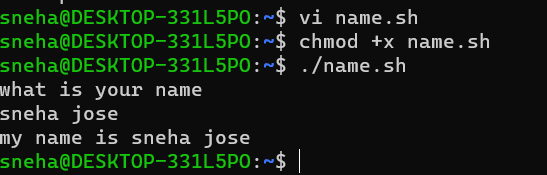
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:

vi

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

$vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

date

ls

pwd

mkdir file1

ls

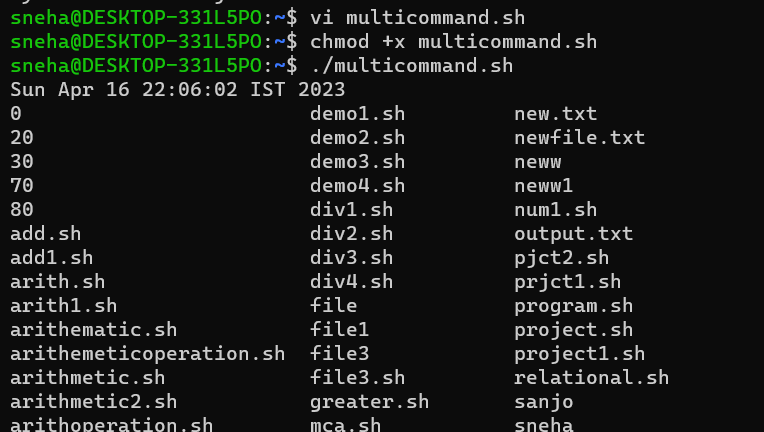
Press ‘Esc’ to end INSERT

: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: $#”

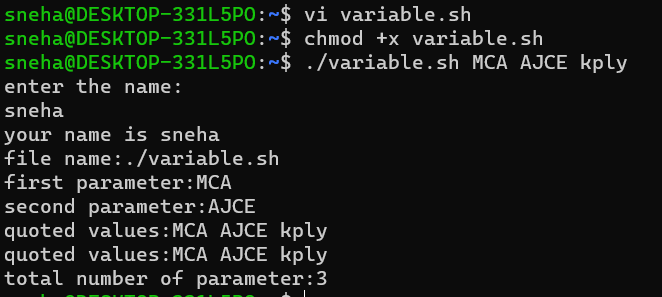
Press ‘Esc’ to end INSERT

: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”

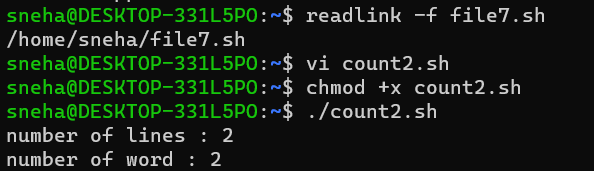
Press ‘Esc’ to end INSERT

: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[0] = “name1”

Name[1] = “name2”

Name[2] = “name3”

Name[3] = “name”

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

value=`expr 2 + 13`

echo "Total value : $value"

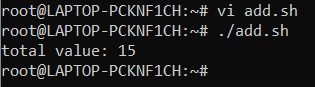
Press ‘Esc’ to end INSERT

: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

num1=12

num2=48

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

echo "Sum : $sum"

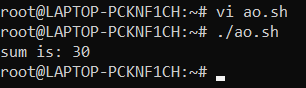
Press ‘Esc’ to end INSERT

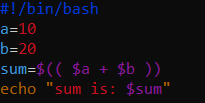
:wq!

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 INSERT

#!/bin/bash

sum=$(( $1 + $2 ))

echo "Sum : $sum"

Press ‘Esc’ to end INSERT

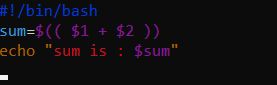
:wq!

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

read -p "Enter the First number: " num1

read -p "Enter the Second number: " num2

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

echo "Sum : $sum"

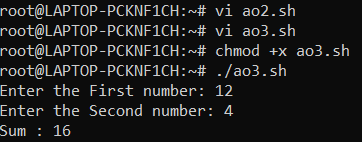
Press ‘Esc’ to end INSERT

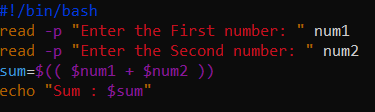
:wq!

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"

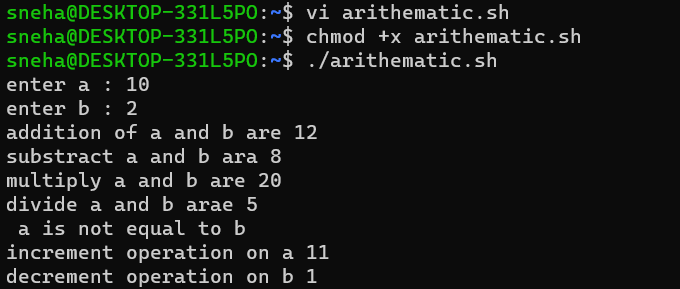
Press ‘Esc’ to end INSERT

: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 the First number: " num1

read -p "Enter the Second number: " num2

if(( $num1 == $num2 ))

then

echo "== : $num1 is equal to $num2"

else

echo "== : $num1 is not equal to $num2"

fi

if(( $num1 != $num2 ))

then

echo "!= : $num1 is not equal to $num2"

else

echo "!= : $num1 is equal to $num2"

fi

if(( $num1 > $num2 ))

then

echo "> : $num1 is greater than $num2"

else

echo "> : $num1 is not greater than $num1"

fi

if(( $num1 < $num2 ))

then

echo "< : $num1 is less than $num2"

else

echo "< : $num1 is not less than $num2"

fi

if(( $num1 >= $num2 ))

then

echo ">= : $num1 is greater than or equal to $num2"

else

echo ">= : $num1 is not greater than or equal to $num2"

fi

if(( $num1 <= $num2 ))

then

echo "<= : $num1 is less than or equal to $num2"

else

echo "<= : $num1 is not less than or equal to $num2"

fi

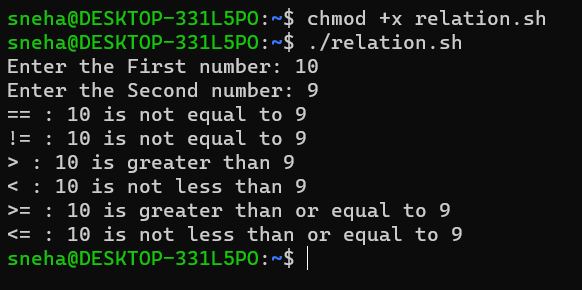
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

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

num1=12

num2=10

if [ $num1 -eq $num2 ]

then

echo "-eq : $num1 is equal to $num2"

else

echo "-eq: $num1 is not equal to $num2"

fi

if [ $num1 -ne $num2 ]

then

echo "-ne: $num1 is not equal to $num2"

else

echo "-ne: $num1 is equal to $num2"

fi

if [ $num1 -gt $num2 ]

then

echo "-gt: $num1 is greater than $num2"

else

echo "-gt: $num1 is not greater than $num1"

fi

if [ $num1 -lt $num2 ]

then

echo "-lt: $num1 is less than $num2"

else

echo "-lt: $num1 is not less than $num2"

fi

if [ $num1 -ge $num2 ]

then

echo "-ge: $num1 is greater than or equal to $num2"

else

echo "-ge: $num1 is not greater than or equal to $num2"

fi

if [ $num1 -le $num2 ]

then

echo "-le: $num1 is less than or equal to $num2"

else

echo "-le: $num1 is not less than or equal to $num2"

fi

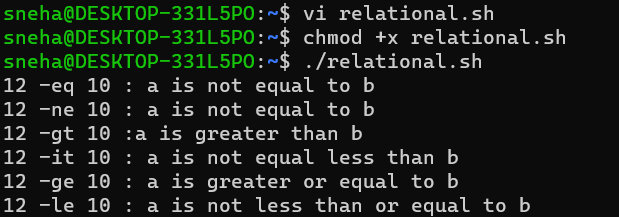
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



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

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the First Boolean: " b1

read -p "Enter the Second Boolean: " b2

if(($b1 == "true" & $b2 == "true" ))

then

echo Both are true

else

echo Both are not true

fi

if(($b1 == "true" || $b2 == "true" ))

then

echo Atleast one of them is true

else

echo None of them are true

fi

if(( ! $b1 == "true" ))

then

echo "b1" was initially false

else

echo "b1" was initially true

fi

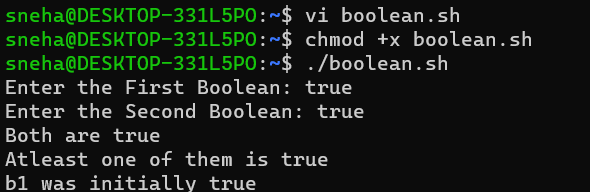
Press ‘Esc’ to end INSERT

: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 a Number: " num1

if(( $num1 == 0 ))

then

echo "$num1 is neither odd nor even number"

elif(( $num1 % 2 == 0 ))

then

echo "$num1 is an even number"

else

echo "$num1 is a odd number"

fi

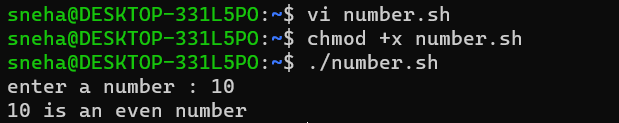
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:

vi

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 a Number: " num1

if(( $num1 > 0 ))

then

echo "$num1 is a positive number"

elif(( $num1 < 0 ))

then

echo "$num1 is a negative number"

else

echo "$num1 is zero"

fi

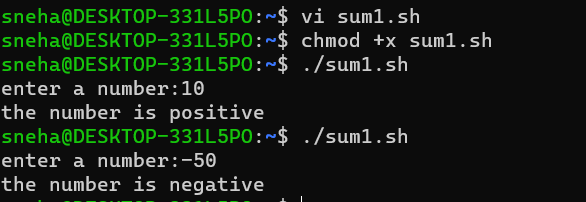
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./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 first number: " num1

read -p "Enter the second number: " num2

if(( $num1 > $num2 ))

then

echo "$num1 is greater than $num2"

else

echo "$num2 is greater than $num1"

fi

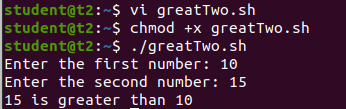
Press ‘Esc’ to end INSERT

: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 the first number: " num1

read -p "Enter the second number: " num2

read -p "Enter the third number: " num3

if(( $num1 > $num2 & $num1 > $num3 ))

then

echo "$num1 is the greatest of the three"

elif(( $num2 > $num3 ))

then

echo "$num2 is the greatest of the three"

else

echo "$num3 is the greatest of the three"

fi

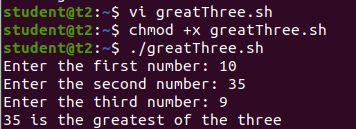
Press ‘Esc’ to end INSERT

: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 first string: " str1

read -p "Enter the second string: " str2

if(( $str1=$str2 ))

then

echo "Both strings are equal"

else

echo "Both strings are not equal"

fi

if(( $str1!=$str2 ))

then

echo "Both strings are not equal"

else

echo "Both strings are equal"

fi

if(( -z$str1 ))

then

echo "String size is zero"

else

echo "String size is non-zero"

fi

if(( -n$str1 ))

then

echo "String size is non-zero"

else

echo "String size is zero"

fi

if(( $str1 ))

then

echo "String is Empty"

else

echo "String is not Empty"

fi

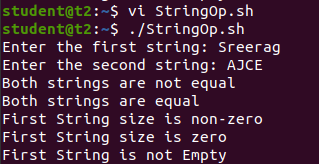
Press ‘Esc’ to end INSERT

: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’ to INSERT

#!/bin/bash

read -p "Enter the first value: " bin1

read -p "Enter the second value: " bin2

result=$(( $bin1&$bin2 ))

echo "Bitwise AND: $result"

result=$(( $bin1|$bin2 ))

echo "Bitwise OR: $result"

result=$(( $bin1^$bin2 ))

echo "Bitwise XOR: $result"

result=$(( ~$bin1 ))

echo "Bitwise COMPLEMENT of First Value: $result"

result=$(( $bin1<<2 ))

echo "Bitwise RIGHT SHIFT(4) of First Value: $result"

result=$(( $bin1>>2 ))

echo "Bitwise LEFT SHIFT(4) of Second Value: $result"

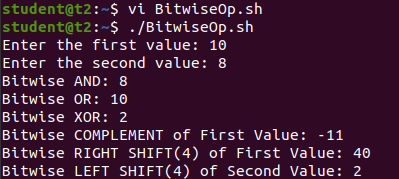
Press ‘Esc’ to end INSERT

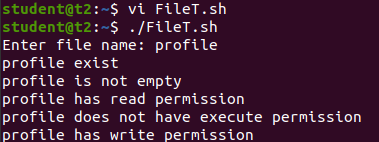
:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell script to demonstrate File Test operators (Exist(e), Size(s), Read Permission(r), Execute Permission(x), Write Permission(w)) by taking user input
   1. vi filename.sh
   2. Press ‘i’ to INSERT
   3. #!/bin/bash
   4. read -p "Enter file name: " f1
   5. if [ -e $f1 ]
   6. then
   7. echo "$f1 exist"
   8. else
   9. echo "$f1 does not exist"
   10. fi
   11. if [ -s $f1 ]
   12. then
   13. echo "$f1 is not empty"
   14. else
   15. echo "$f1 is empty"
   16. fi
   17. if [ -r $f1 ]
   18. then
   19. echo "$f1 has read permission"
   20. else
   21. echo "$f1 does not have read permission"
   22. fi
   23. if [ -x $f1 ]
   24. then
   25. echo "$f1 has execute permission"
   26. else
   27. echo "$f1 does not have execute permission"
   28. fi
   29. if [ -w $f1 ]
   30. then
   31. echo "$f1 has write permission"
   32. else
   33. echo "$f1 does not have write permission"
   34. fi
   35. Press ‘Esc’ to end INSERT
   36. :wq!
   37. chmod +x filename.sh
   38. ./filename.sh
   39. Output:
   40. 
2. 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

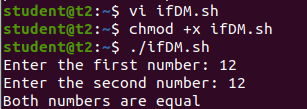
Press ‘Esc’ to end INSERT

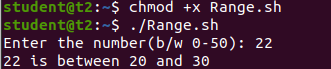
:wq!

chmod +x filename.sh

./filename.sh

Output:



1. Shell Script to check the range of a number if numbers using else if ladder
   1. vi filename.sh
   2. Press ‘i’ to INSERT
   3. #!/bin/bash
   4. read -p "Enter the number(b/w 0-50): " num1
   5. if(( $num1>=0&&$num1<=10 ))
   6. then
   7. echo "$num1 is between 0 and 10"
   8. elif(( $num1>=11&&$num1<=20 ))
   9. then
   10. echo "$num1 is between 10 and 20"
   11. elif(( $num1>=21&&$num1<=30 ))
   12. then
   13. echo "$num1 is between 20 and 30"
   14. elif(( $num1>=31&&$num1<=40 ))
   15. then
   16. echo "$num1 is between 30 and 40"
   17. elif(( $num1>=41&&$num1<=50 ))
   18. then
   19. echo "$num1 is between 40 and 50"
   20. fi
   21. Press ‘Esc’ to end INSERT
   22. :wq!
   23. chmod +x filename.sh
   24. ./filename.sh
   25. Output:
   26. 
2. Shell Script to display the grade of a student by accepting his mark.
   1. A