**Experiment No.: 1**

**Aim**

Introduction to computer hardware. Study of various peripherals. Study of common operating systems. File system organization in common operating systems.

**CO1**

Install and configure common operating systems.

**Procedure**

1. Motherboard:

A motherboard is the main circuit board inside a computer that connects all of the computer's hardware components together, such as the central processing unit (CPU), memory, storage drives, and expansion cards. It serves as a communication hub between these components, allowing them to work together and perform various functions.

The motherboard typically includes several important components, including the chipset and the BIOS (Basic Input/Output System). Modern motherboards come in different form factors, such as ATX, microATX, and mini-ITX, which determine their size and layout.

Overall, the motherboard is a critical component that plays a central role in the operation of a computer.

1. RAM:

RAM (Random Access Memory) modules are computer components that store data and instructions temporarily while the computer is running. RAM is a type of volatile memory, which means that its contents are erased when the computer is turned off or restarted.

RAM modules come in various types, speeds, and capacities. They are typically installed in slots on the motherboard and can be easily upgraded or replaced. RAM modules are also used to support multitasking, where multiple programs can run simultaneously.

RAM modules are an essential component of a computer that provides temporary storage for data and instructions. They help to improve the computer's performance and support multitasking.

1. Daughter Card:

A daughter card, also known as a daughterboard or expansion card, is a circuit board that connects to the main motherboard of a computer to add new functionality or enhance existing features.

Daughter cards are commonly used to expand the capabilities of a computer, such as adding additional ports, memory, or processing power. Examples of daughter cards include graphics cards, sound cards, network interface cards (NICs), and storage expansion cards.

1. Bus Slot:

A bus slot, also known as an expansion slot, is a socket on the motherboard of a computer that allows expansion cards to be inserted and connected to the computer's bus system.

There are several types of bus slots commonly used in computers, including Peripheral Component Interconnect (PCI), PCI Express (PCIe), and Accelerated Graphics Port (AGP) slots. These slots vary in their bandwidth, power, and physical size, and are designed to accommodate different types of expansion cards.

Expansion cards, such as graphics cards, sound cards, and network interface cards, are connect to the computer's bus system through the bus slot, allowing them to communicate with other components and exchange data.

1. SMPS:

SMPS stands for Switched-Mode Power Supply, and it is a type of power supply used in computers and other electronic devices. The SMPS is responsible for converting AC power from a wall outlet into DC power that the computer can use to operate. SMPS uses high-frequency switching and regulation to convert AC power to DC power more efficiently.

SMPS units are widely used in modern computers, as they are more reliable, energy-efficient, and generate less heat than linear power supplies.

SMPS is an essential component of modern computers that plays a vital role in powering the system and ensuring its proper operation.

1. Internal Storage Devices:

Internal storage devices are electronic components used to store data within a computer or other electronic devices. There are two main types of internal storage devices: Hard Disk Drives (HDDs) and Solid State Drives (SSDs).

HDDs are the traditional type of internal storage device and are typically larger in capacity than SSDs. They consist of spinning disks that store data magnetically and read/write heads that move over the disks to access the data.

SSDs, on the other hand, use flash memory to store data and have no moving parts, which makes them faster, more durable, and more energy-efficient than HDDs. They are typically more expensive than HDD.

1. Interfacing Ports:

Interfacing ports refers to the process of connecting different devices or components together in order to exchange information or perform certain tasks. Ports are used to facilitate communication between devices, and they can come in different forms, such as USB ports, Ethernet ports, HDMI ports, and so on.

Interfacing ports is an important aspect of modern computing, as it allows us to connect a wide range of devices and components together in order to create complex systems and networks. Whether it's connecting a keyboard and mouse to a computer, or setting up a network of servers and workstations in a large enterprise, interfacing ports is a crucial part of the process.

**Experiment No.: 2**

**Aim**

Familiarization of Linux Commands.

**CO2**

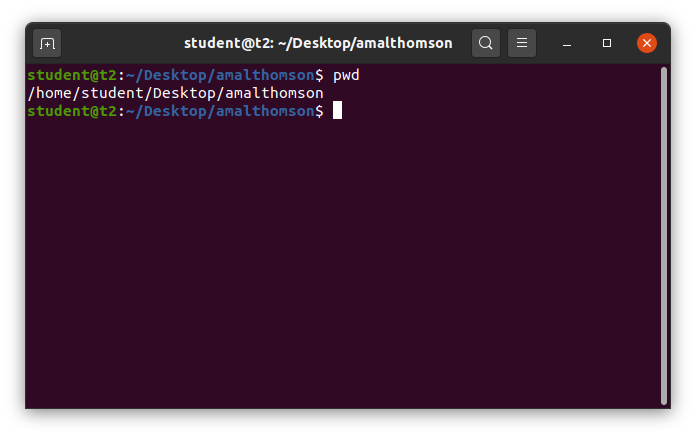
Perform system administration tasks.

**Procedure**

1. **pwd** – used to print the working directory. After execution it shows the absolute path.

Syntax: $ pwd

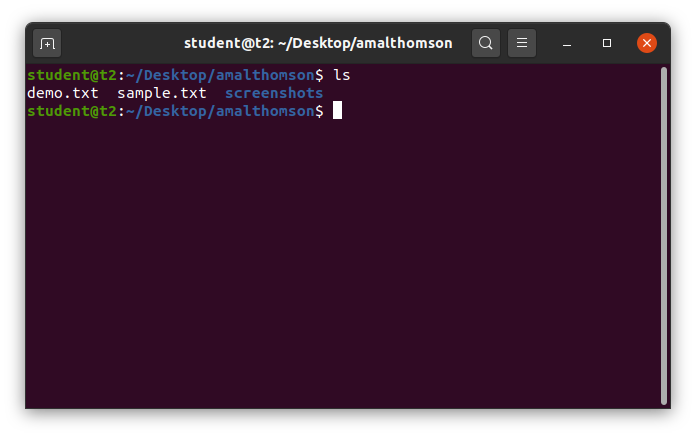
Output:



1. **ls** – used to list the files and content in the directory.

Syntax: $ ls

Output:

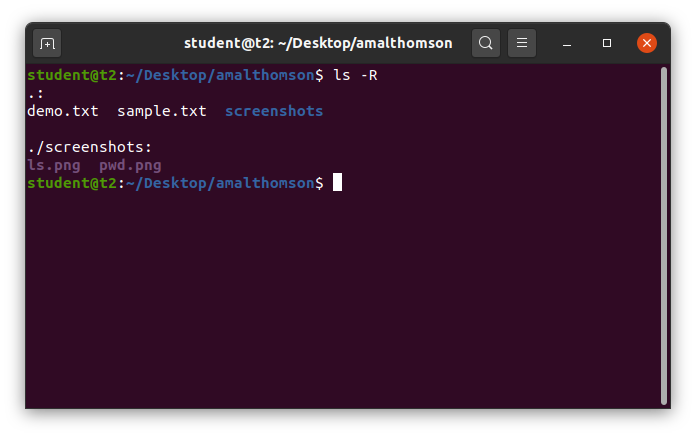


**Options of ls command.**

1. **ls -R** – used to list the directory as well as the subdirectory.

Syntax: $ ls -R

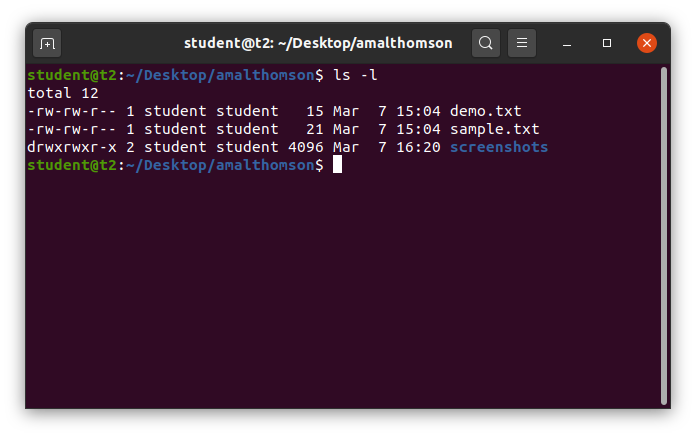
Output:



1. **ls -l** – used to view the long list of directory.

Syntax: $ ls -l

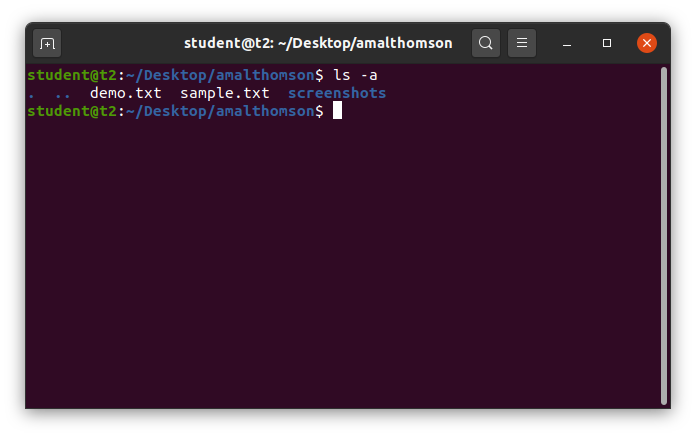
Output:



1. **ls -a** – used to view the list in directory along with hidden files.

Syntax: $ ls -a

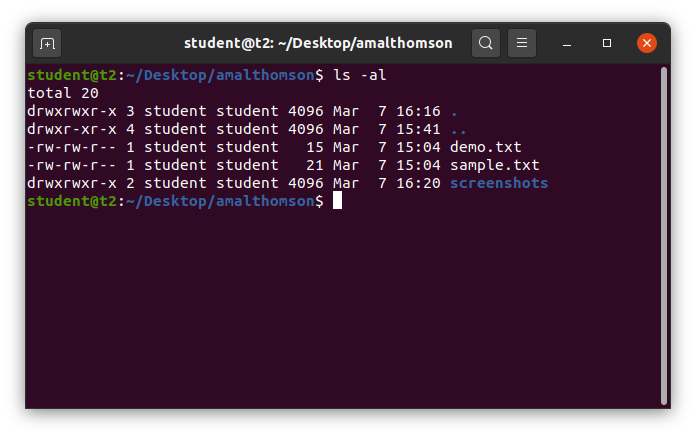
Output:



1. **ls -al** – used to view the list in directory with detailed information along with hidden files.

Syntax: $ ls -al

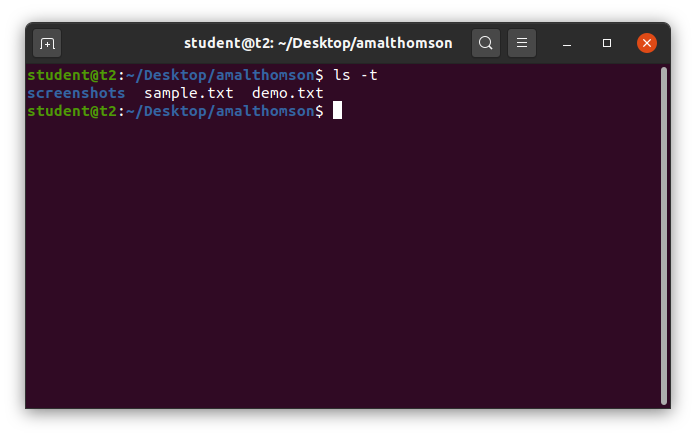
Output:



1. **ls -t** – used to view the list in sorted order of last modified.

Syntax: $ ls -t

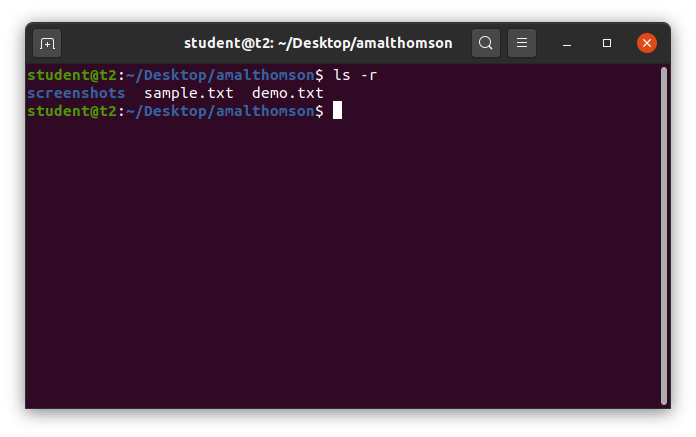
Output:



1. **ls -r** – used to view the list in reverse order of last modified.

Syntax: $ ls -r

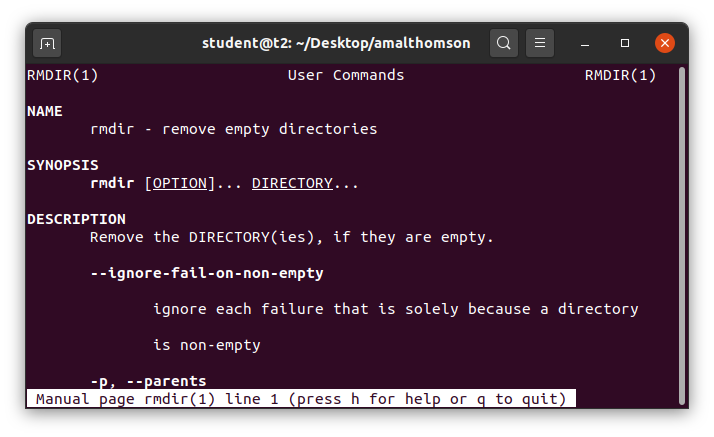
Output:



1. **man** - used to learn and understand the existing commands we can learn and understand about different commands from the shell using man command.

Syntax: $ man mkdir

Output:



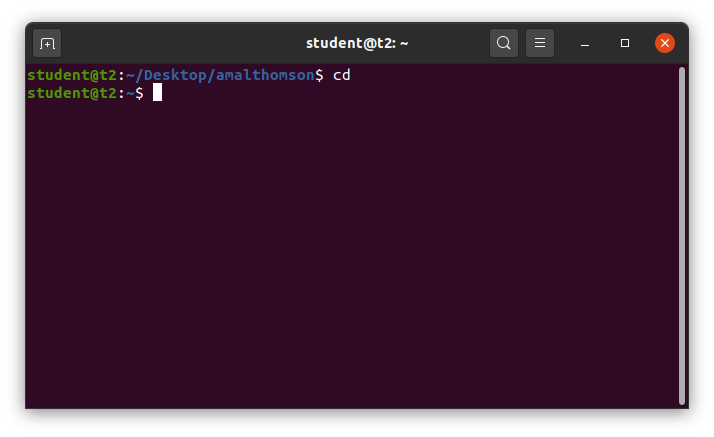
1. **cd** – used to navigate through directory.

**Options of cd commands**:

1. cd – used to switch to home directory.

Syntax: $ cd

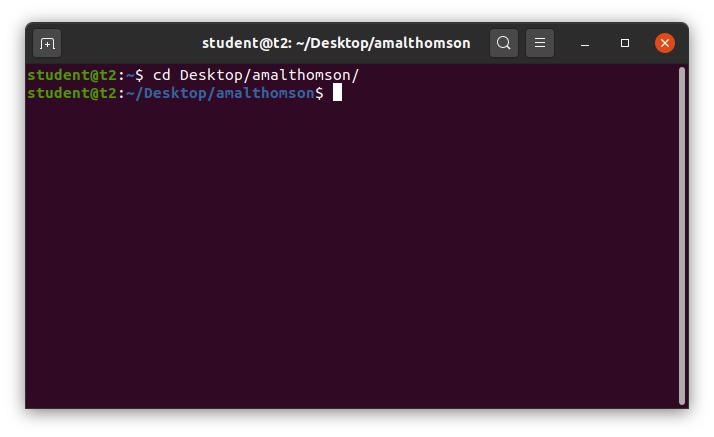
Output:



1. **cd <path>** - used to change to a particular path or directory

Syntax: $ cd *<directory\_path>*

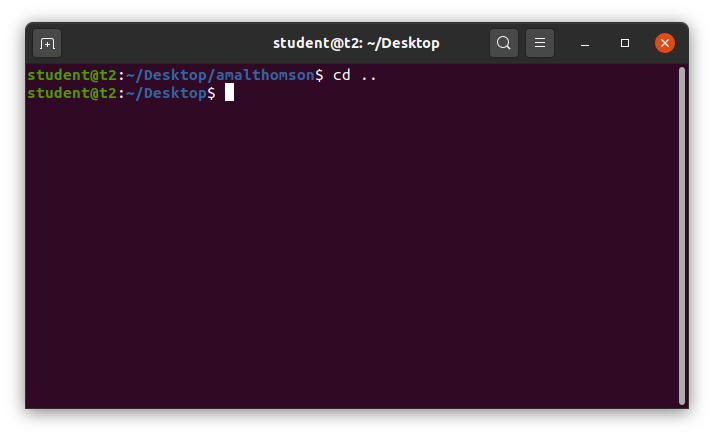
Output:



1. **cd ..** – used to switch back to previous directory or one directory back from the current directory

Syntax: $ cd ..

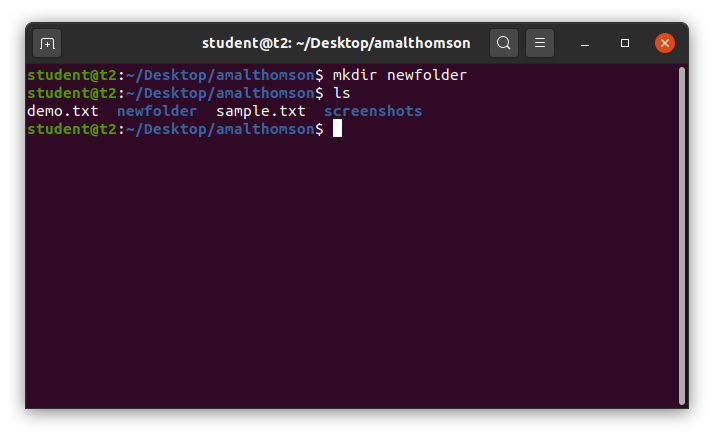
Output:



1. **mkdir** – Used to make new directory.

Syntax: $ mkdir *<directory\_name>*

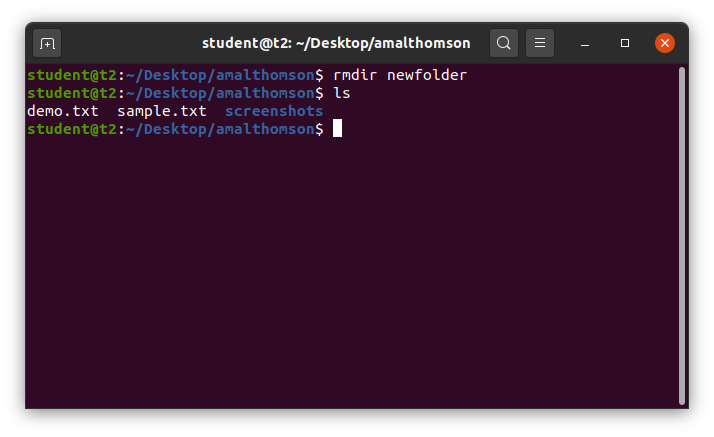
Output:



1. **rmdir** – used to remove a directory.

Syntax: $ rmdir <*directory\_name*>

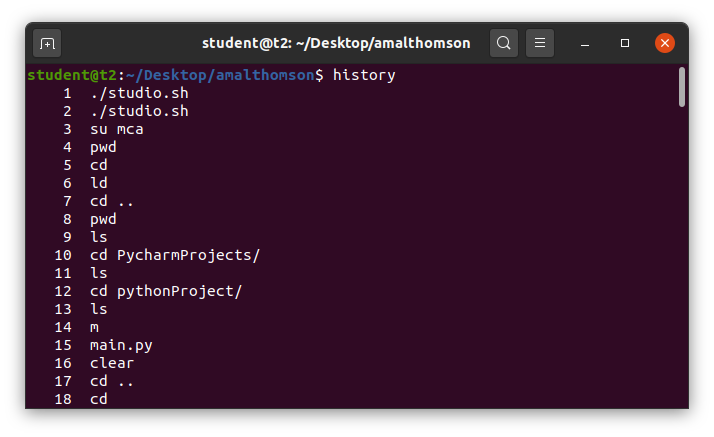
Output:



1. **history** – used to view the list of commands executed in a certain period of time.

Syntax: $ history

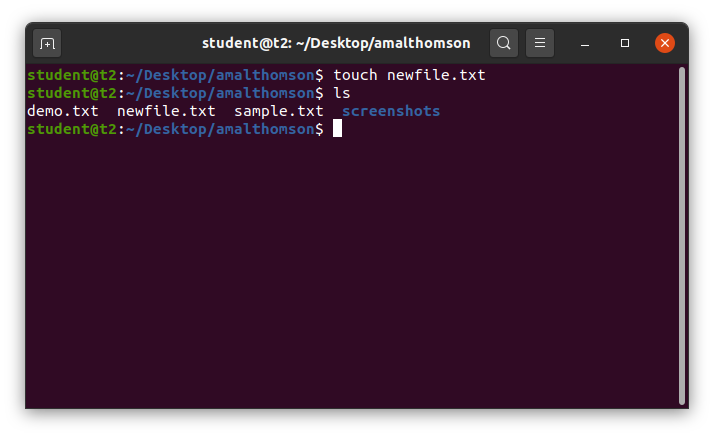
Output;



1. **touch** – used to create a new blank file.

Syntax: $ touch <*filename*>

Output:

****

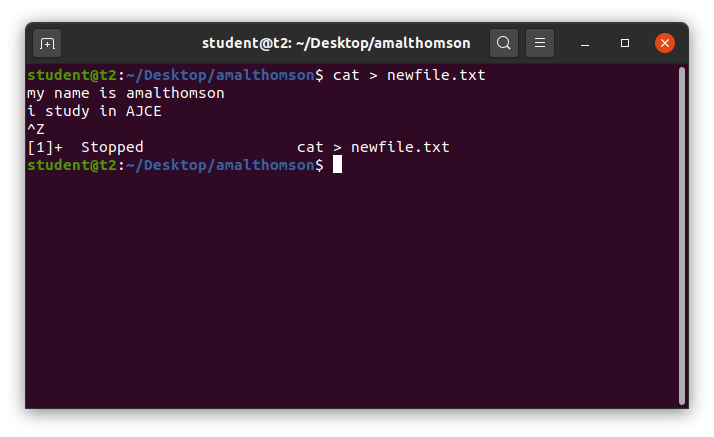
1. **cat –** usedto create a new blank file and also to add contents to the file.

**Options of cat commands:**

1. **cat >** – used to create a new blank file and also to add contents to the file.

Syntax: $ cat > <*filename*>

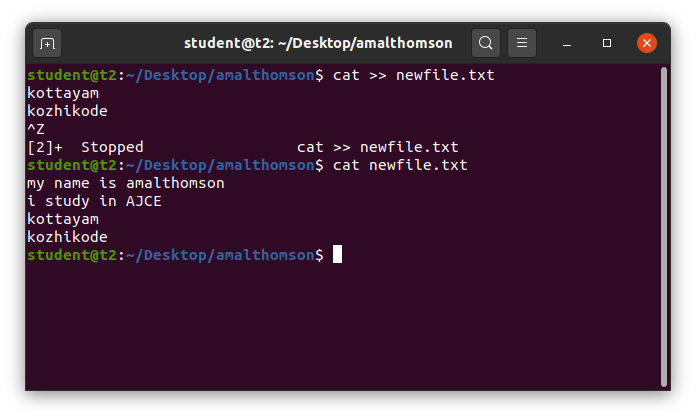
Output:



1. **cat >>** – used to append new contents to existing file.

Syntax: $ cat >> <*filename*>

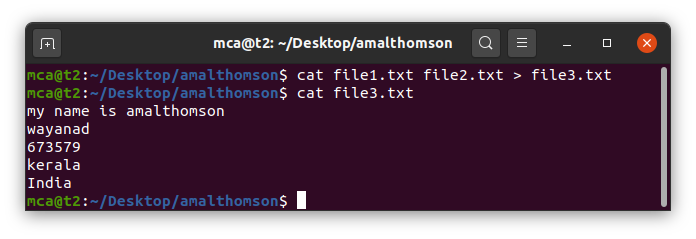
Output:



1. **cat file1 file2 > file3** – copy contents of two files to a third file.

Syntax: $ cat *<filename><filename> > <filename>*

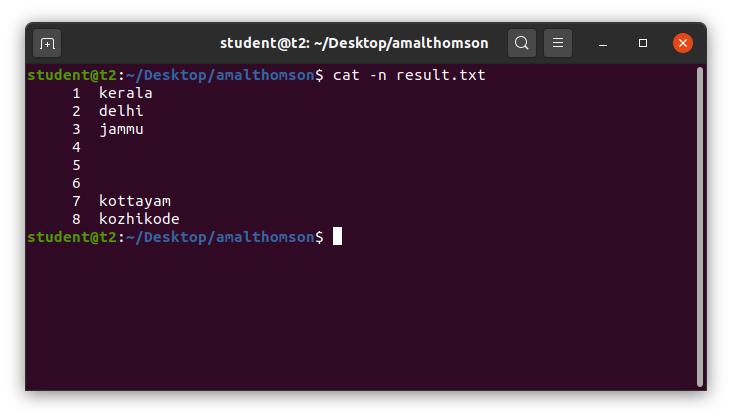
Output:



1. **cat -n** – to display the contents with line numbers.

Syntax: $ cat -n <*filename*>

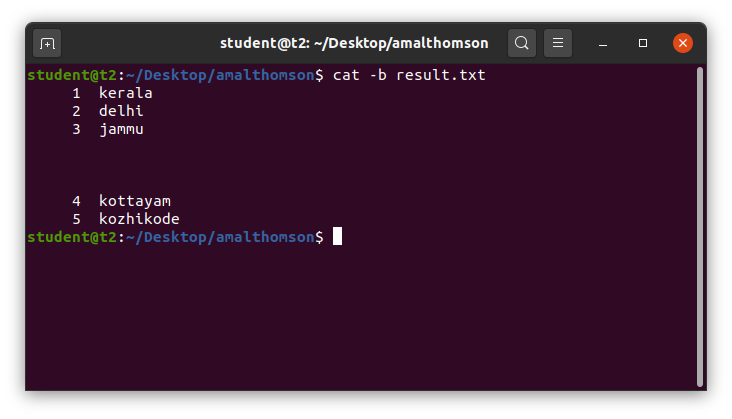
Output:



1. **cat -b** – to remove numbering for empty lines.

Syntax: $ cat -n <*filename>*

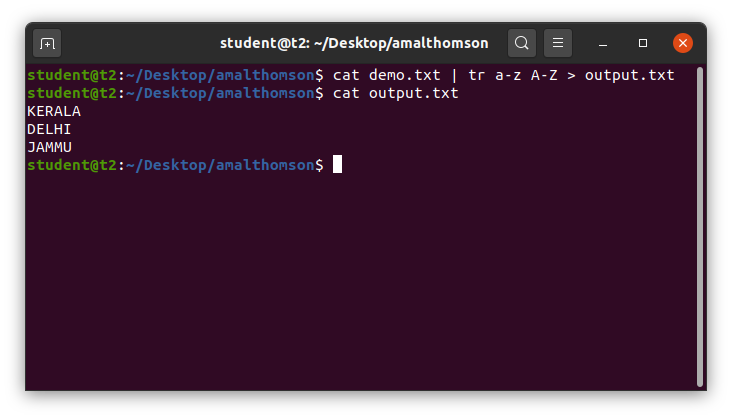
Output:



1. **cat *<filename> |* tr a-z A-Z > <*filename>*** – converts the contents of a file to UpperCase and saves into another file.

Syntax: $ cat <*filename>* | tr a-z A-Z > <*filename*>

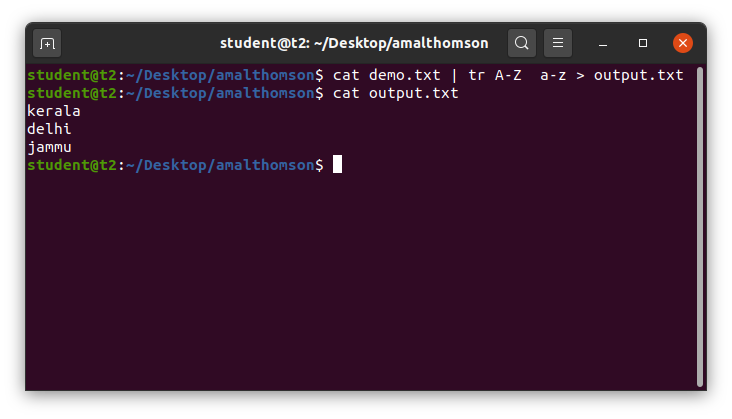
Output:



1. **cat *<filename> |* tr A-Z a-z > <*filename>*** – converts the contents of a file to LowerCase and saves into another file.

Syntax: $ cat <*filename>* | tr A-Z a-z > <*filename*>

Output:



**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

**Experiment No.: 3**

**Aim**

Familiarization of Linux Commands.

**CO2**

Perform system administration tasks.

**Procedure**

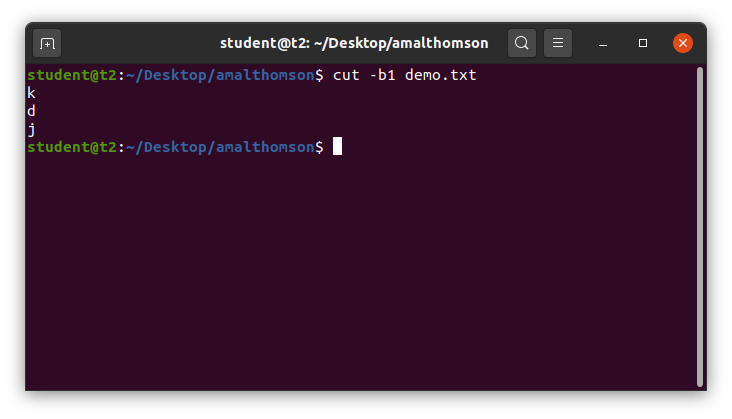
1. **cut** – to cut the contents of the file.

**Options of cut command**:

1. **cut -b1** – to cut the contents of a file by byte position.

Syntax: $ cut -b1 <*filename*>

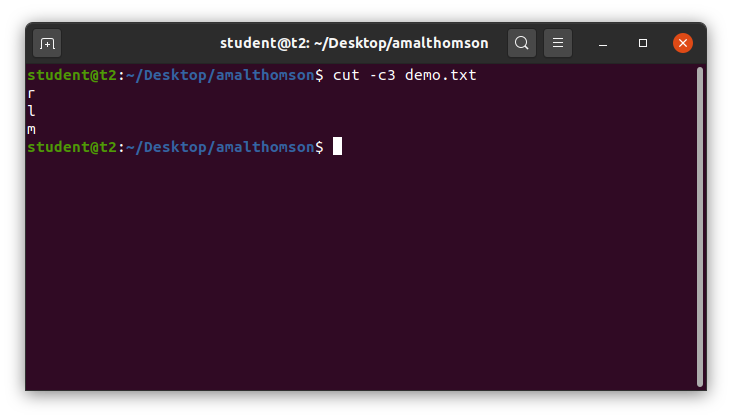
Output:



1. **cut -c3** – to cut the contents of a file by character position.

Syntax: $ cut -c3 <*filename*>

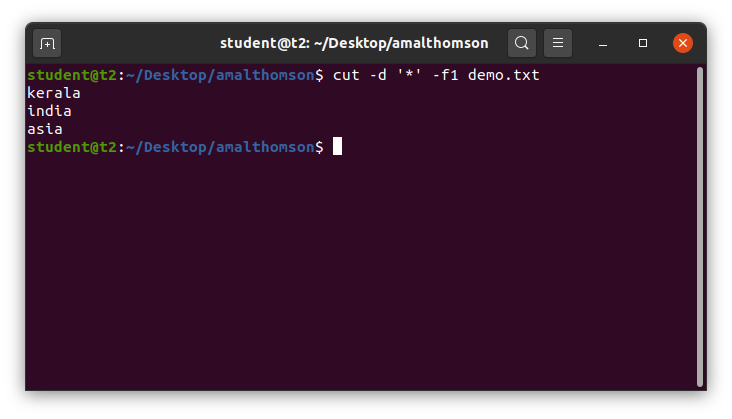
Output:



1. **cut –d ‘\*’ -f1** – use delimiter to cut the contents at ‘\*’ in the first column which is given by –f1.

Syntax: $ cut –d ‘\*’ -f1 <*filename*>

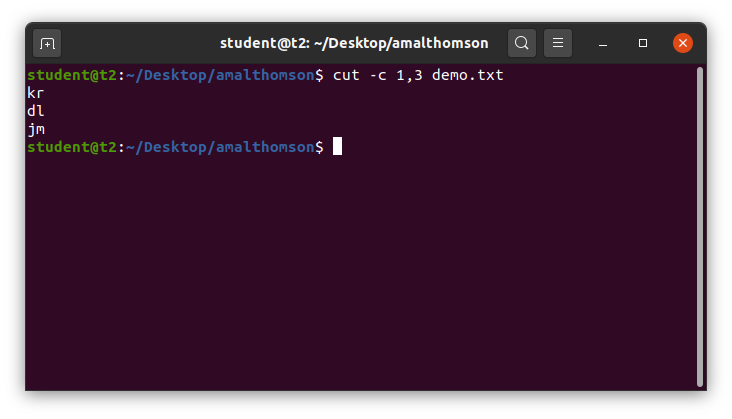
Output:



1. **cut –c** – to cut the characters from a specified position in a file.

Syntax: $ cut -c [1,3] <*filemane*>

Output:



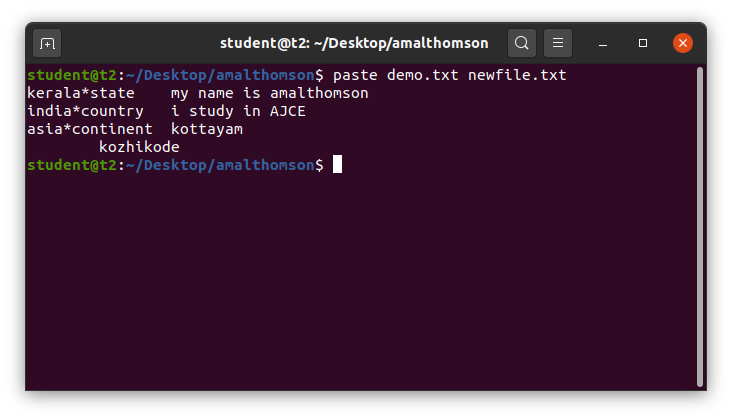
1. **paste** – to paste the content of a file to another.

**Options of paste command**

1. **paste <*filename*> <*filename*>** – to paste the contents in file1 to file2.

Syntax: $ paste <*filename*> <*filename*>

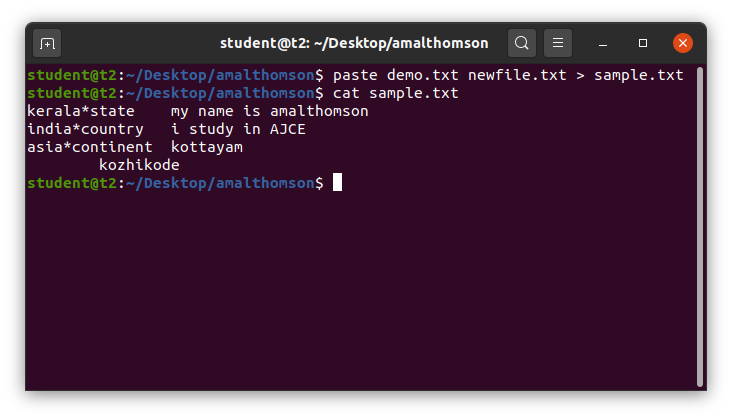
Output:



1. **paste *<filename> <filename> > <filename>*** *–* to paste the contents of two files to a third file.

Syntax: $ paste *<filename> <filename> > <filename>*

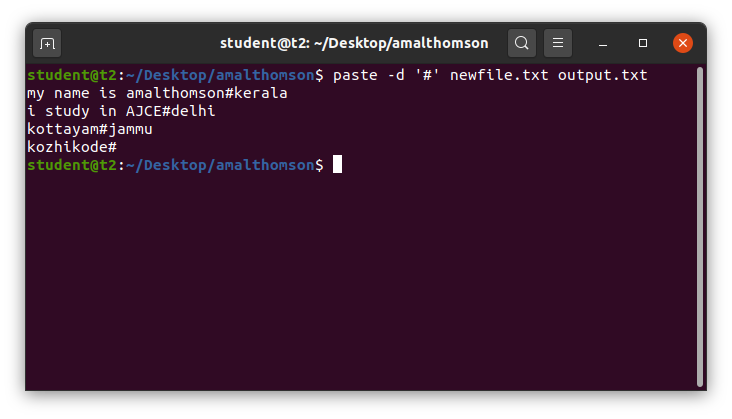
Output:



1. **paste –d ‘#’ <filename> <filename>** – to paste # and join the contents of a file with another file.

Syntax: $ paste –d ‘#’ *<filename> <filename>*

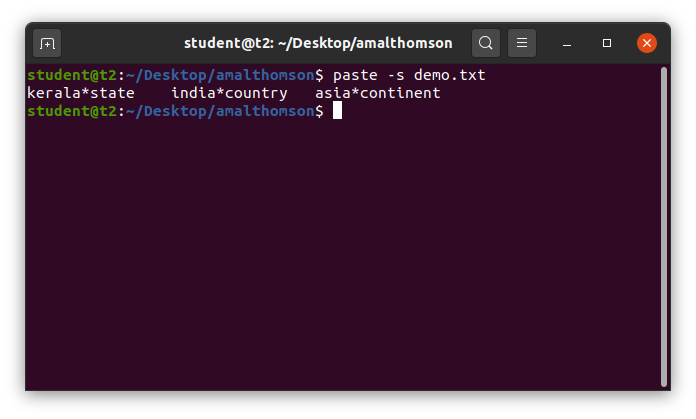
Output:



1. **paste -s** – to show all contents of a file in a single line.

Syntax: $ paste -s <*filename*>

Output:

****

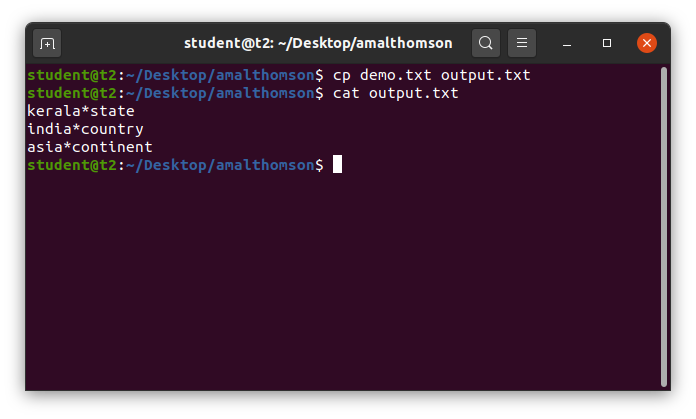
1. **cp** – to copy the contents of a file.

Options of cp command:

1. **cp *<filename> <filename>*** *–* to copy the contents of a file into another file or a new file.

Syntax: $ cp *<filename> <filename>*

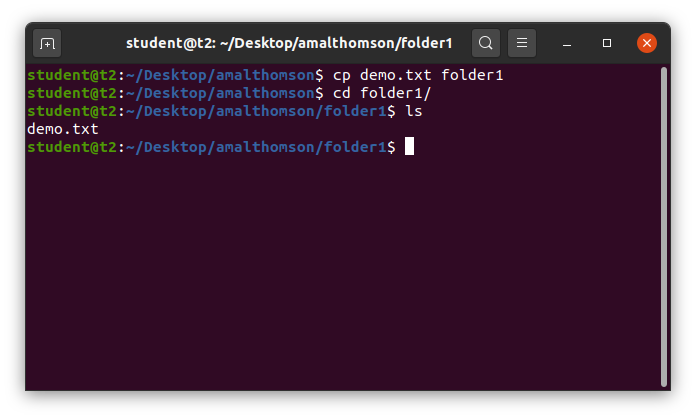
Output:



1. **cp <*filename*> <*directory*>** - to copy a file to a directory.

Syntax: $ cp <*filename*> <*directory*>

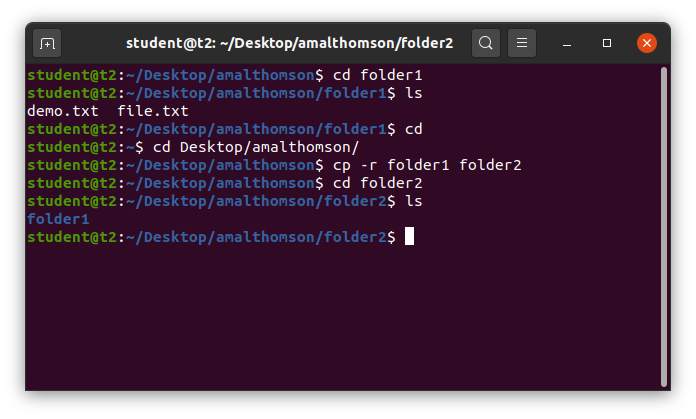
Output:



1. **cp -r** - to copy a directory and its contents to another directory.

Syntax: $ cp -r <*directory*> <*directory*>

Output:



**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

**Experiment No.: 4**

**Aim**

Familiarization of Linux Commands.

**CO2**

Perform system administration tasks.

**Procedure**

1. **read** – to read the contents of a line into a variable.

**Options of read command**

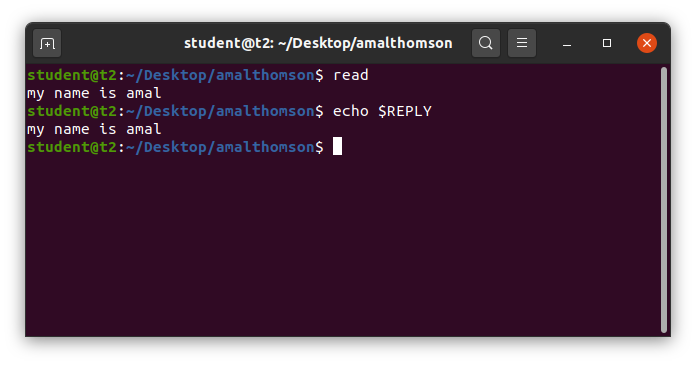
1. **read** – read contents of a line into variable.

Syntax: $ read

My name is amal

$ echo $REPLY

Output:



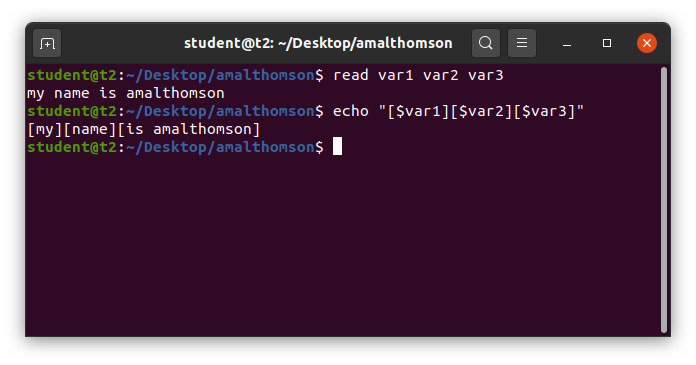
1. **read <*variable\_name*>** – read contents of a line to a particular variables.

Syntax: $ read var1 var2 var3

My name is amalthomson

$ echo “[var1][var2][var3]”

Output:



1. **read** – read from multiple lines

Syntax:

$ read

my \

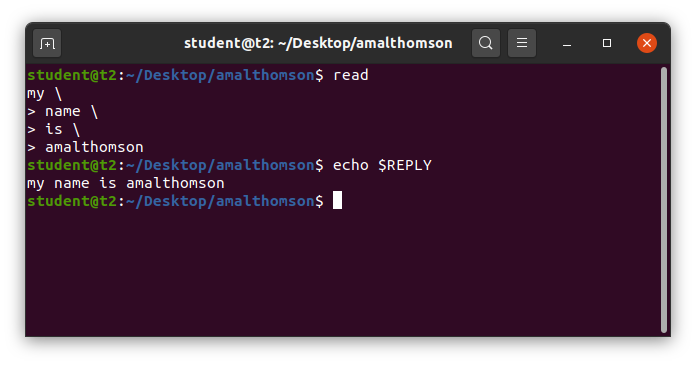
name \

is \

amalthomson

$ echo $REPLY

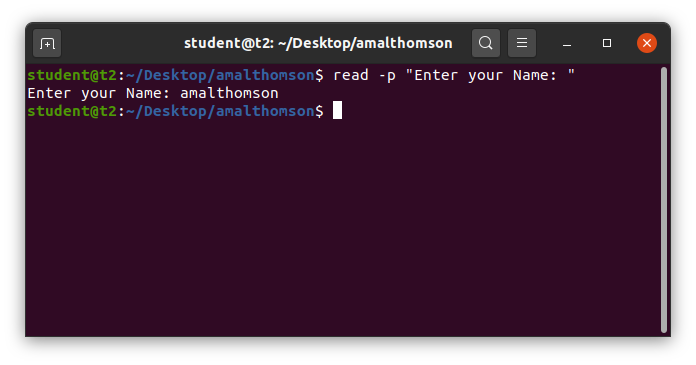
Outout:



1. **read -p** – read with prompt message

Syntax: $ read -p “Enter your name”

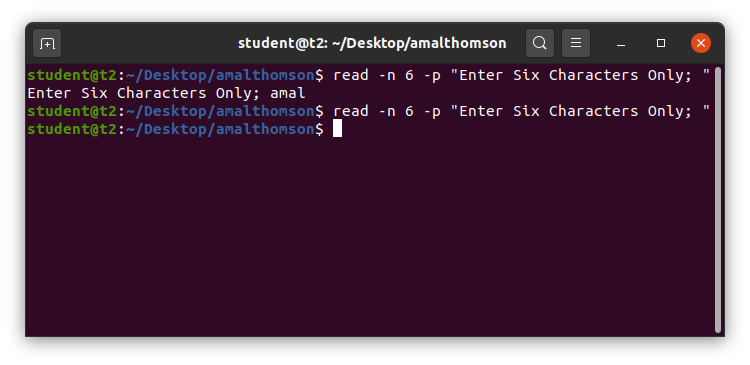
Output:



1. **read -n** – read with limit of characters can be read

Syntax: $ read -n -p “Enter only six characters”

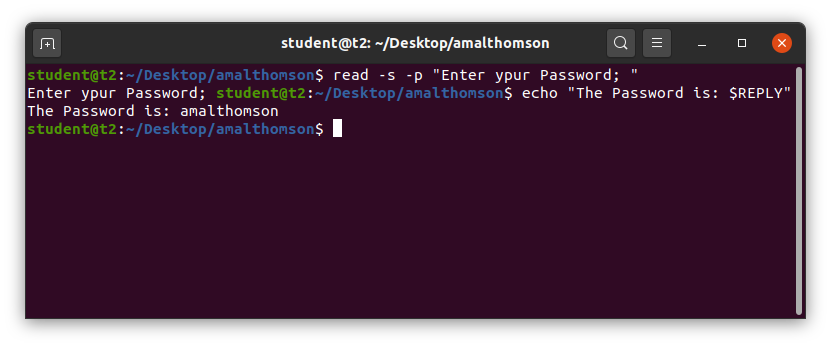
Output:



1. **read -s** – read lines securely without displaying the data entered

Syntax: $ read -s -p “Enter your password”

Output:



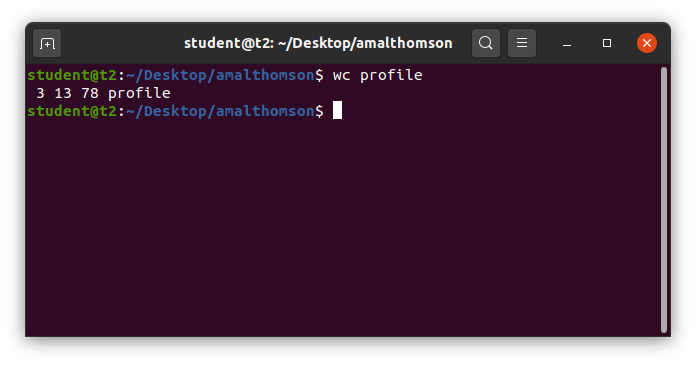
1. **wc** – word count

**Options of wc commands**

1. **wc <*filename*>** – to display number of lines, words, bytes and filename from a file

Syntax: $ wc profile.txt

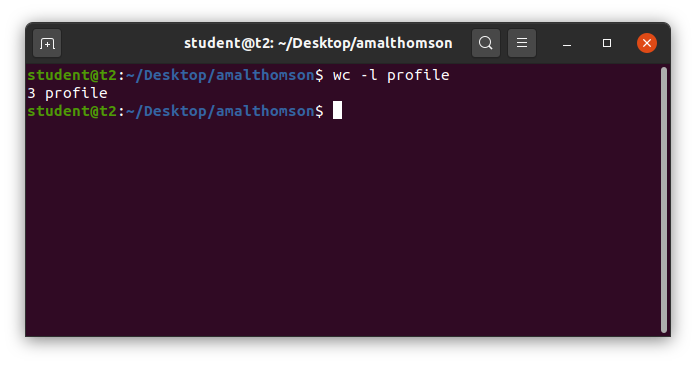
Output:

****

1. **wc -l <*filename*>** – to display number of lines and filename from a file

Syntax: $ wc -l profile.txt

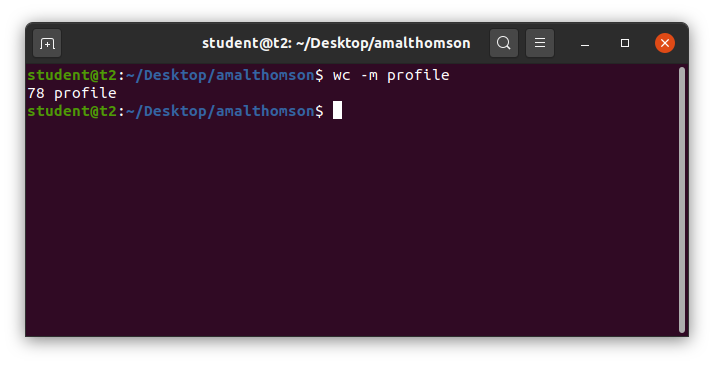
Output:



1. **wc -m <*filename*>** – to display number of bytes and filename from a file

Syntax: $ wc -m profile.txt

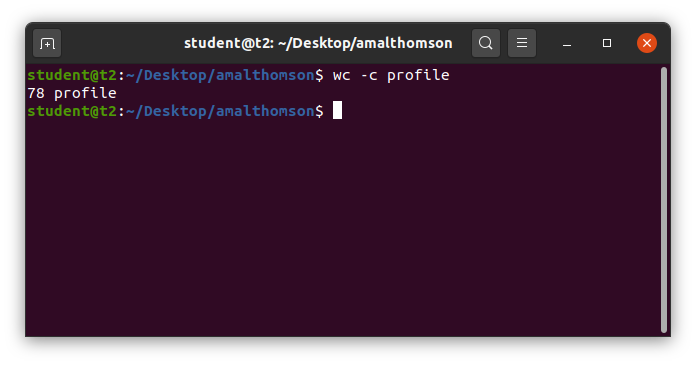
Output:



1. **wc -c <*filename*>** – to display number of characters and filename from a file

Syntax: $ wc -c profile.txt

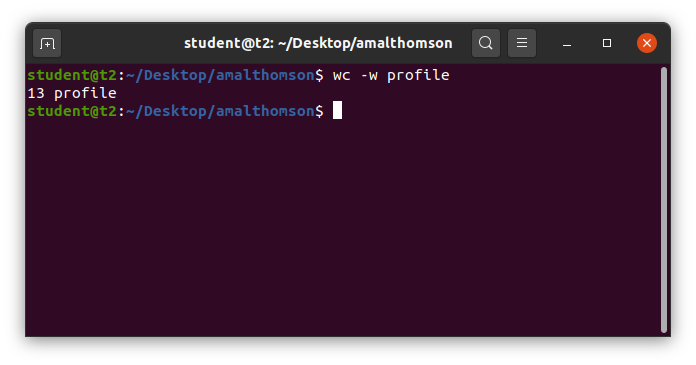
Output:



1. **wc -w <*filename*>** – to display number of words and filename from a file

Syntax: $ wc -w profile.txt

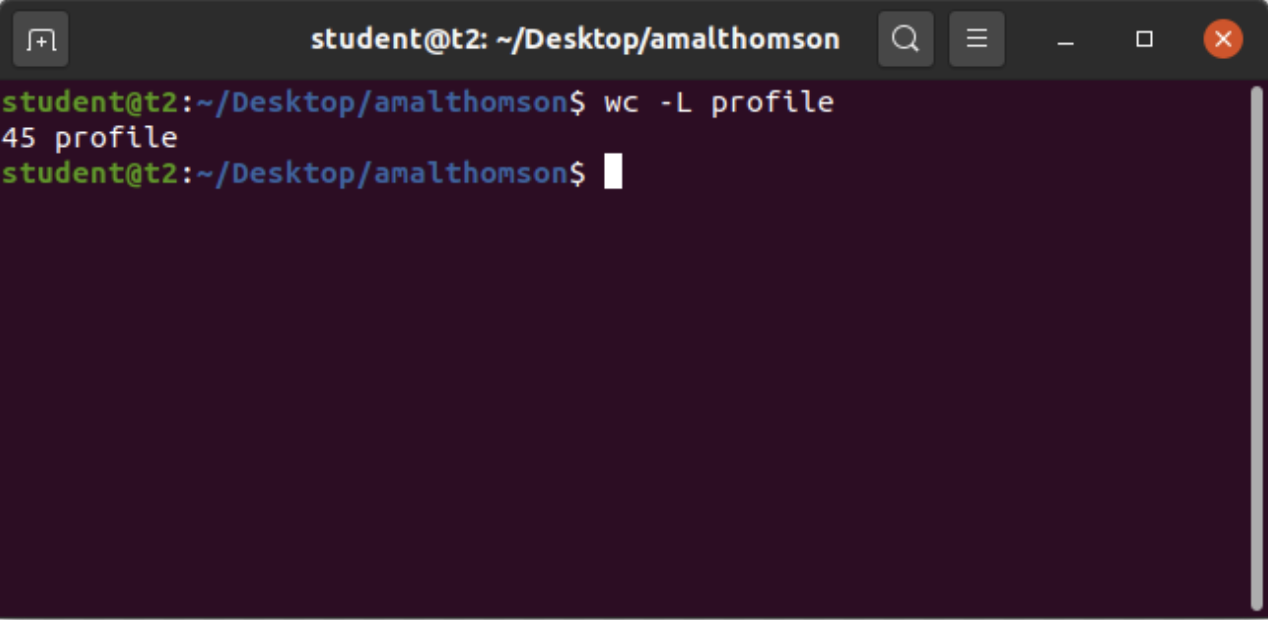
Output:



1. **wc -L <*filename***> – to display length of largest line.

Syntax: $ wc -L profile.txt

Output:



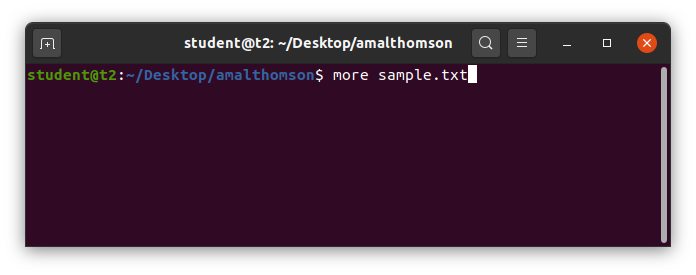
1. **more** – is similar to get to display the contents, the only difference is that in case of longer text or content get command output will scroll off your screen while more command display the output only screen full at a time.

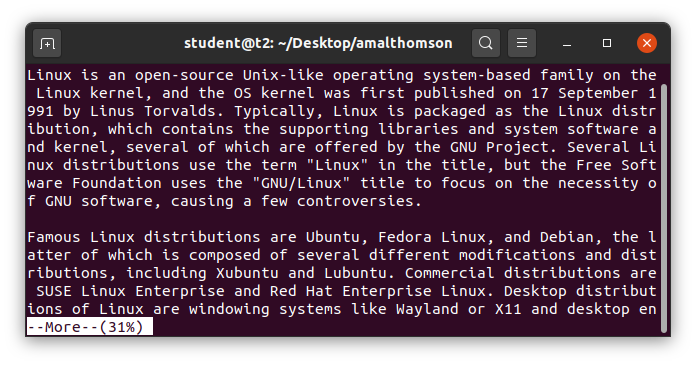
Options of more command

1. **more <*filename*>** – display contents of a file

Syntax: $ more sample.txt

Output:

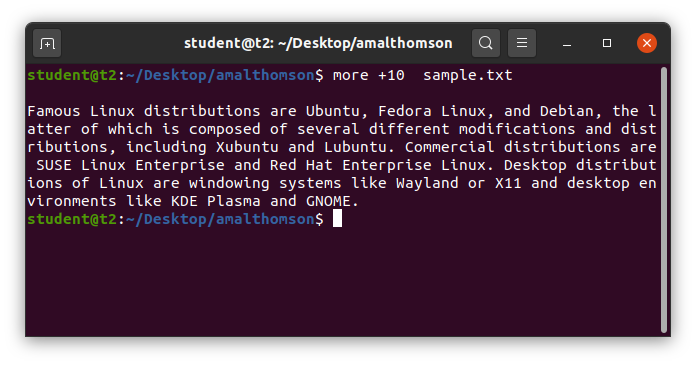




1. **more +20 <*filename*>** – display contents of a file

Syntax: $ more +10 sample.txt

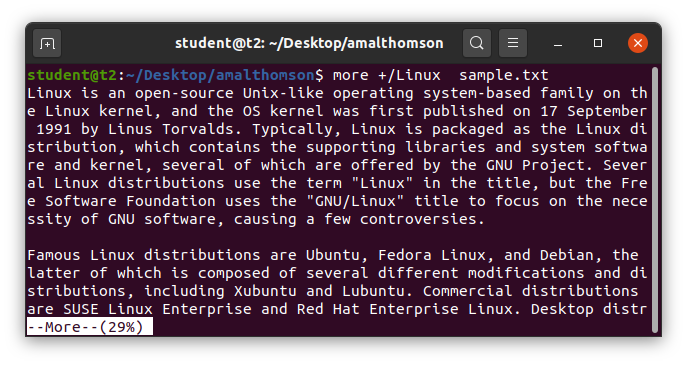
Output:



1. **more +/pattern <*filename***> – to search is train inside your document, you can view all the instances by navigating through the result

Syntax: $ more +/Linux sample.txt

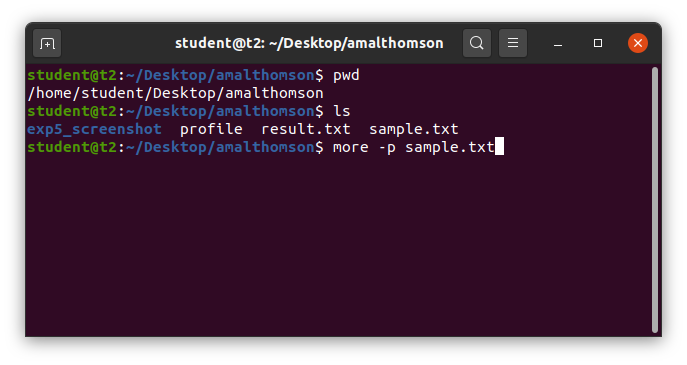
Output:

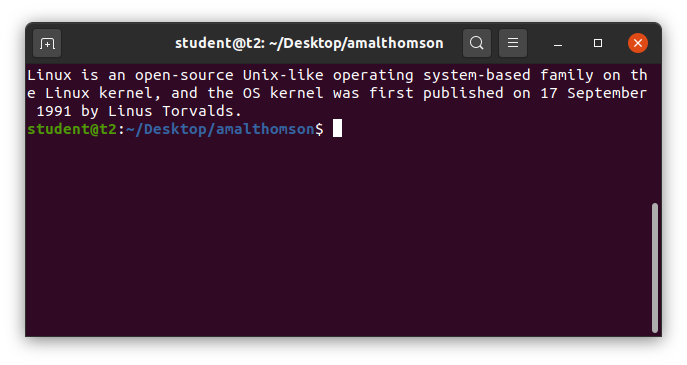


1. **more -p <*filename*>** – to display the contents of a file after clearing the screen

Syntax: $ more -p sample.txt

Output:

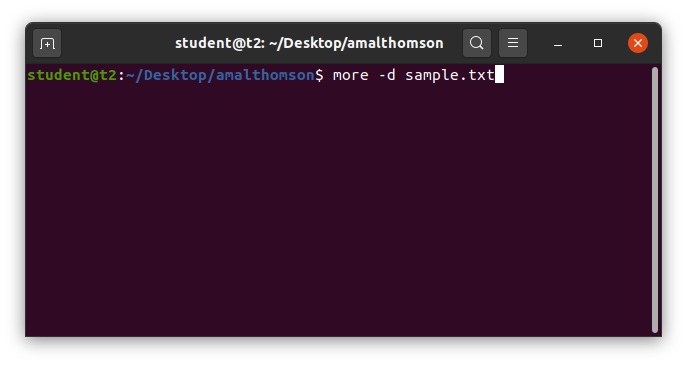


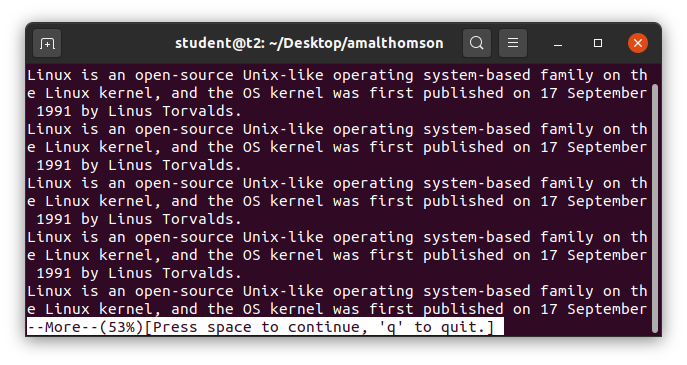


1. **more -d <*filename***> – display instructions such as, space to continue and q to quit.

Syntax: $ more -d sample.txt

Output:





**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

**Experiment No.: 5**

**Aim**

Familiarization of Linux Commands.

**CO2**

Perform system administration tasks.

**Procedure**

1. grep – used to filter the contents of a file, which makes search easy.

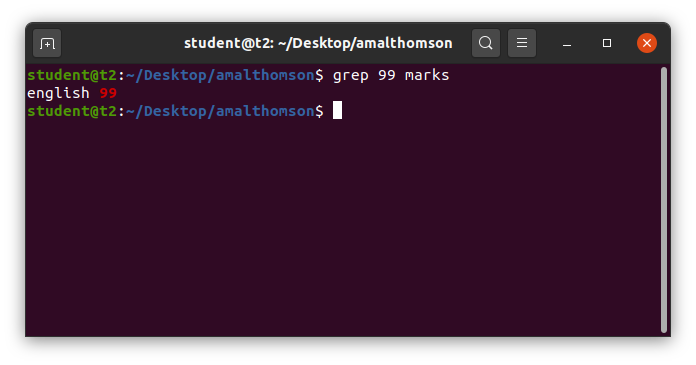
Options of grep command

1. grep <content><filename> – search and display a particular content from a file

Syntax:

$ grep 90 marks

Output:

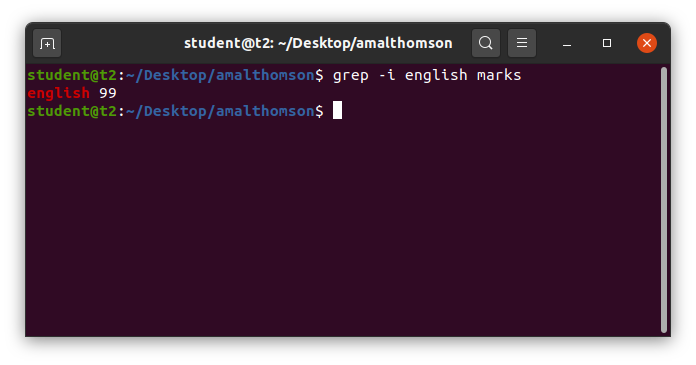


1. grep -i <pattern><filename> – used to search and display a matching pattern, case insensitive

Syntax:

$ grep -i english marks

Output:

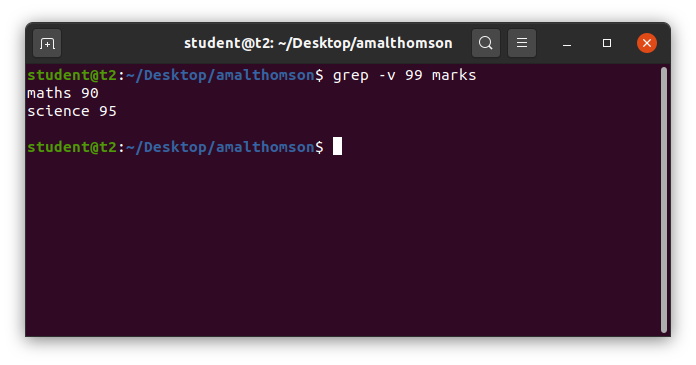


1. grep -v <content><filename> – inverted search and display

Syntax:

$ grep -v 99 marks

Output:

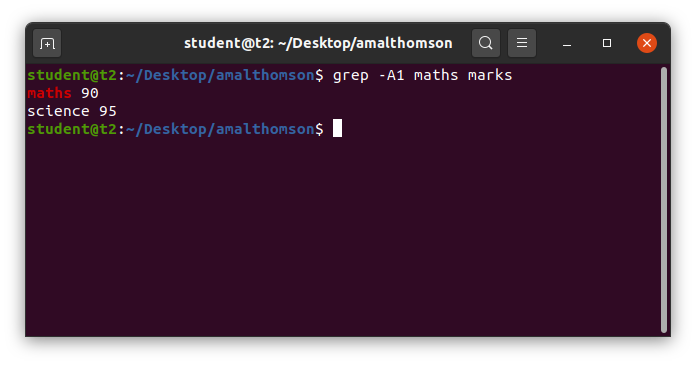


1. grep -A1 <content><filename> – display searched content and the next line from a file.

Syntax:

$ grep -A1 maths marks

Output:

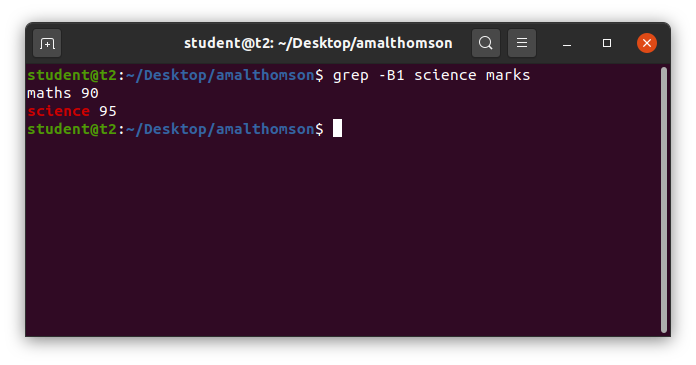


1. grep -B1 <content><filename> – display searched content and the previous line from a file.

Syntax:

$ grep -B1 science marks

Output:

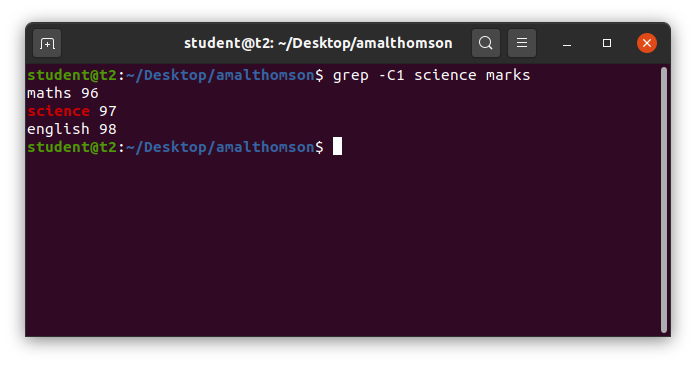


1. grep -C1 <content><filename> – display searched content and the previous and next line from a file.

Syntax:

$ grep -C1 science marks

Output:



1. head – display top contents of the file, by default it displays top 10 lines.

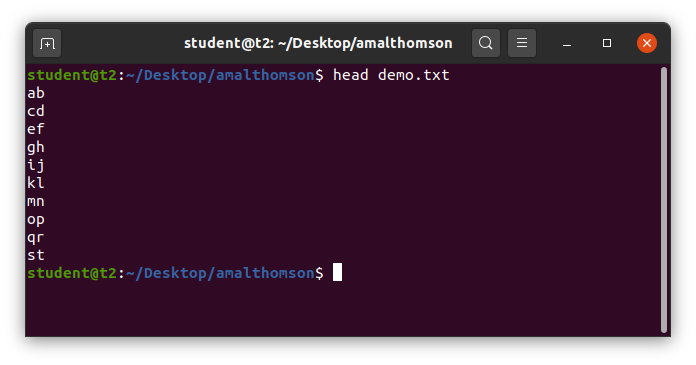
Options of head command.

1. head <filename> – display top 10 lines of a file

Syntax:

$ head demo.txt

Output:

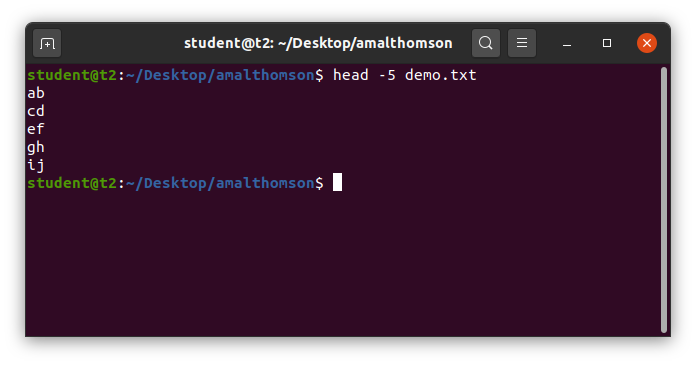


1. head -<limit> <filename> – display top number of lines mentioned in the limit of a file

Syntax:

$ head -5 demo.txt

Output:



1. tail – display bottom contents of the file, by default it displays bottom 10 lines.

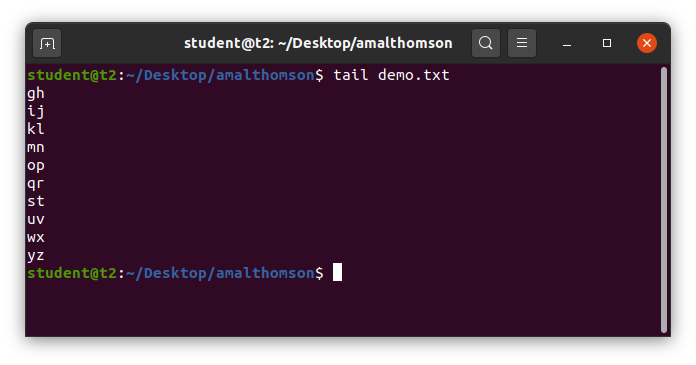
Options of tail command.

1. tail <filename> – display bottom 10 lines of a file

Syntax:

$ tail demo.txt

Output:

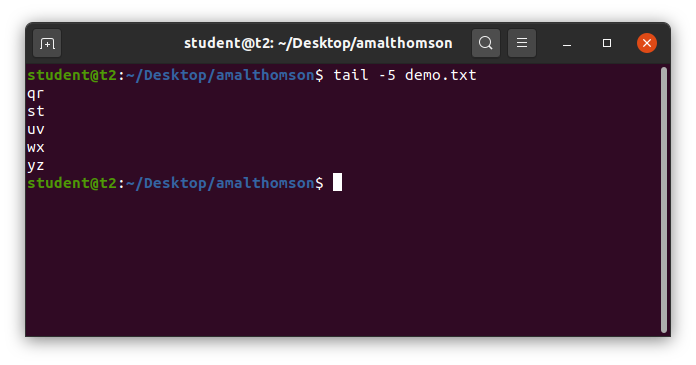


1. tail -<limit> <filename> – display bottom number of lines mentioned in the limit of a file

Syntax:

$ tail -5 demo.txt

Output:



1. mv – used to move files or folders.

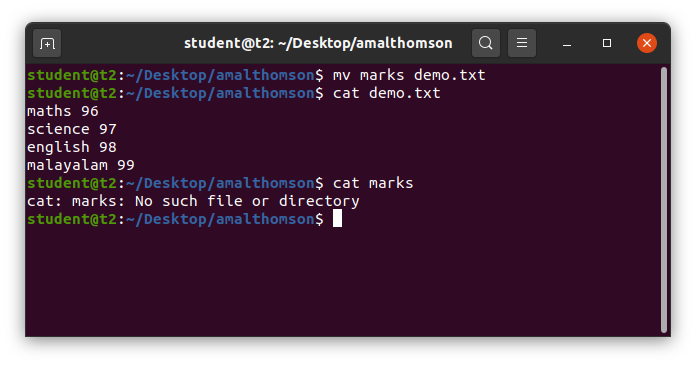
Options of move command

1. mv <filename> <filename> – replaces file2 with file1

Syntax:

$ mv marks demo.txt

Output:

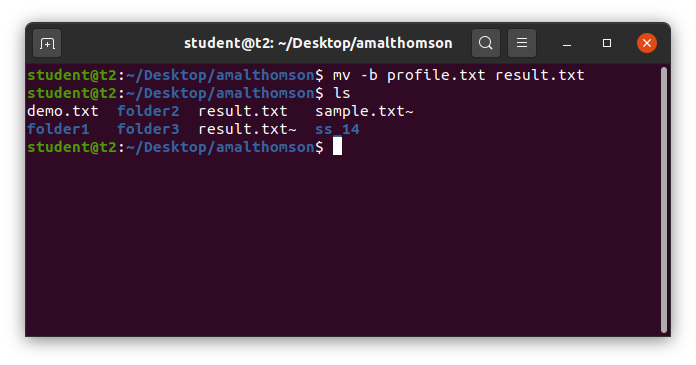


1. mv -b <filename> <filename> – replace file2 with file1 and keeps a backup of the file replaced

Syntax:

$ mv -b profile.txt result.txt

Output:

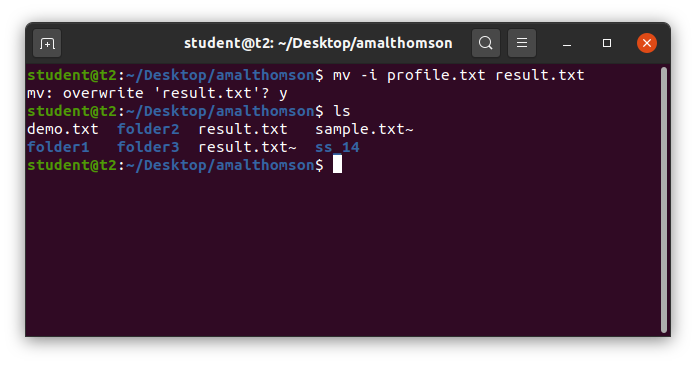


1. mv -i <filename> <filename> – displays a prompt message to confirm overwrite.

Syntax:

$ mv -i profile.txt result.txt

Output:



**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

**Experiment No.: 6**

**Aim**

Familiarization of Linux Commands.

**CO2**

Perform system administration tasks.

**Procedure**

1. expr – evaluates and displays output.

Options of expr:

1. expr – addition

Syntax: $ expr <value> + <value>

1. expr – subtraction

Syntax: $ expr <value> – <value>

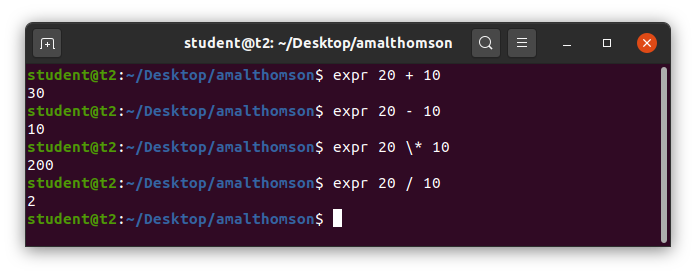
1. expr – multiplication

Syntax: $ expr <value> \\* <value>

1. expr – division

Syntax: $ expr <value> / <value>

Output:



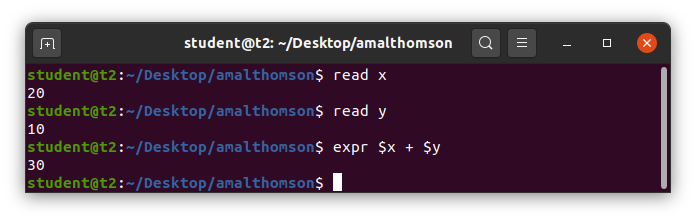
1. expr – read from console

Syntax: $ read x

$ read y

$ expr $x + $y

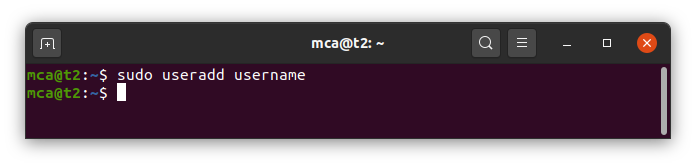
Output:



1. useradd – create a new user

Syntax: $ sudo useradd <username>

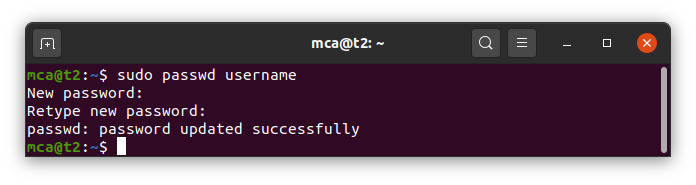
Output:



1. passwd – create/update a password for a user

Syntax: $ sudo passwd <username>

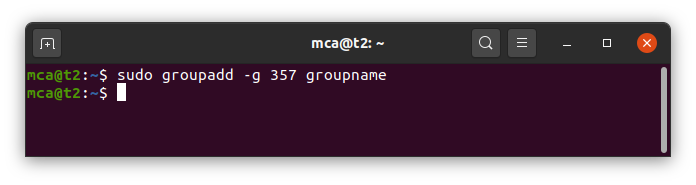
Output:



1. groupadd – create a new group

Synatx: $ sudo groupadd -g <identifier> <groupname>

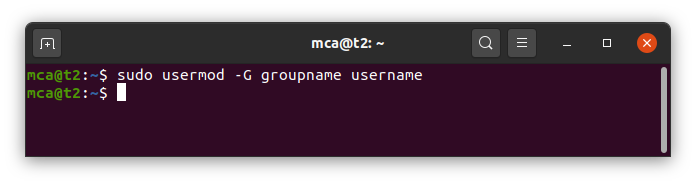
Output:



1. usermod – add a user to the group

Synatx: $ sudo usermod -G <groupname> <username>

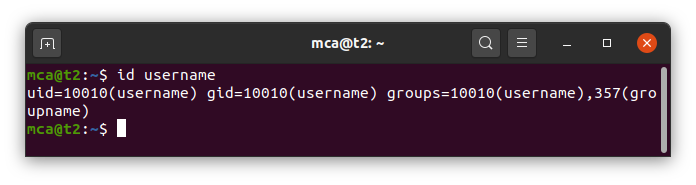
Output:



1. id – returns the details of the user

Syntax: $ id <username>

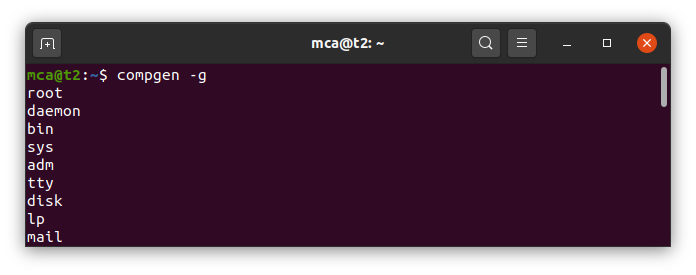
Output:



1. compgen – display all groups

Syntax: $ compgen -g

Output;



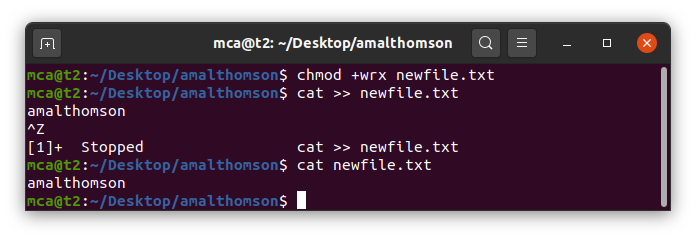
1. chmod – change permisions

Options of chmod:

1. chdmod + – add/allow permissions

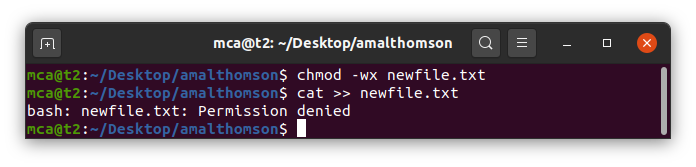
Syntax: $ chmod + wxr <filename>

Output:



1. chmod – – remove/revoke permission

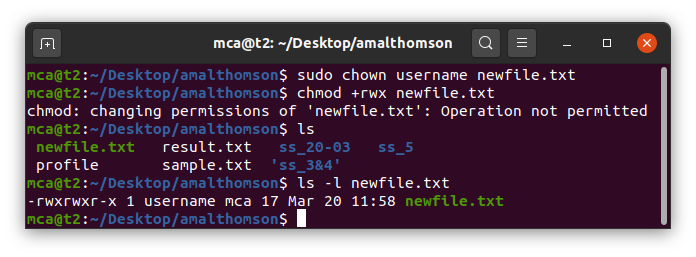
Syntax: $ chmod – wx <filename>



1. chown – change owner of the file/folder

Syntax: $ chown <usrname> <filename>

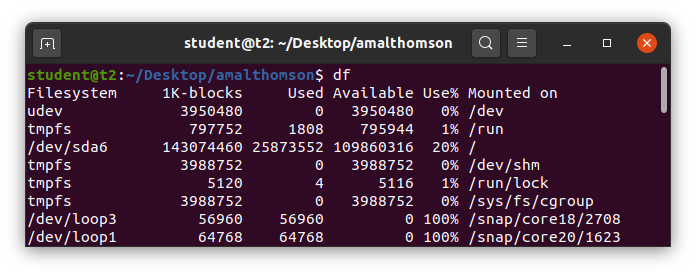
Output:



1. df – display disk utilization of entire disk

Syntax: $ df

Output:



1. du – display disk utilization of a particular file/folder

Syntax: $ du <filename>

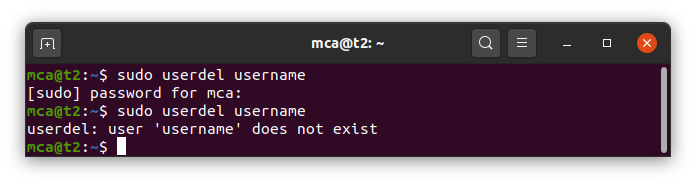
Output:



1. userdel – delete a user

Syntax: $ userdel <username>

Output:



**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

**Experiment No.: 7**

**Aim**

Familiarization of Linux Commands.

**CO2**

Perform system administration tasks.

**Procedure**

1. ip addr – display ip address of a system

Syntax: $ ip addr

Output:

1. ssh <username> @ <ip\_address> – ssh stands for secure shell protocol, used to securely connect to a remote server or system. ssh is secure in the sense that transfers data in encrypted form between the host to the client.

Syntax: $ ssh mca @ 192.168.6.15

Output:

1. $ sudo apt-get install openssh - server - update port – updating port 22, if already taken.

Syntax: $ sudo apt-get install openssh -server – update port

Output:

1. $ sudo ufw allow 22 – allowing port 22 for ssh

Syntax: $ sudo ufw allow 22

Output:

1. $ ssh mca @ 192.168.6.15 – connecting to remote server

Syntax: $ ssh mca @ 192.168.6.15

Output:

1. ssh – keygen – generate a key for secure shell.

Syntax: $ ssh – keygen

Output:

1. ps – stands for process, displays currently running programs or instances.

Options of ps command:

1. ps – displays currently running programs or instances

Syntax: $ ps

Output:

1. ps – C – display is running processes of a specific program.

Syntax: $ ps – C firefox

Output:

1. ps – f – p <process\_id> – list of process by process ID.

Syntax: $ ps – f – p 2276

Output:

1. pu – u – displays all the running processes of a specific user.

Syntax: $ pu – u <username>

Output:

**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained.