

Experiment No.: 3

Aim

Familiarization of Linux Commands.

CO2

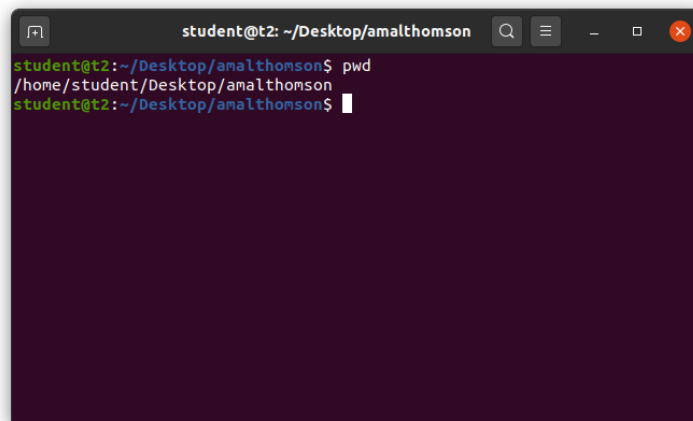
Perform system administration tasks.

Procedure

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

Syntax: \$ pwd

Output:

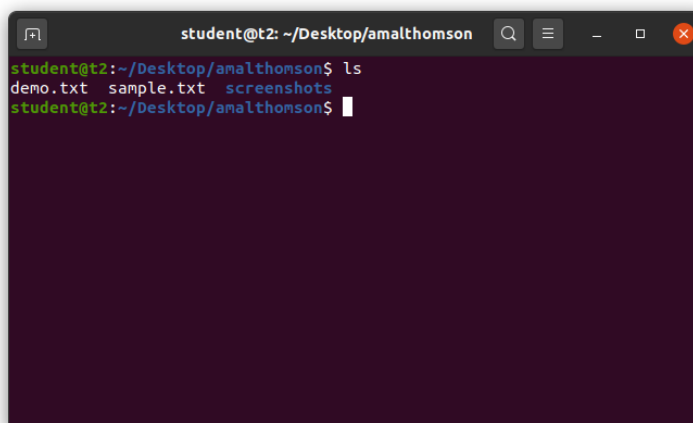


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ pwd
/home/student/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ ls

Output:



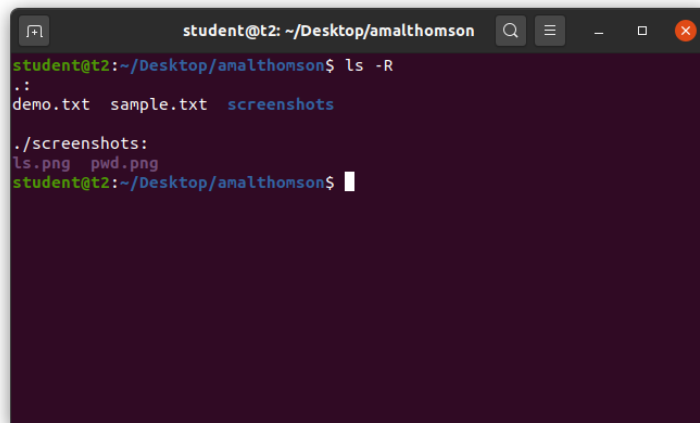
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls
demo.txt  sample.txt  screenshots
student@t2:~/Desktop/amalthomson$
```

Options of ls command.

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

Syntax: \$ ls -R

Output:



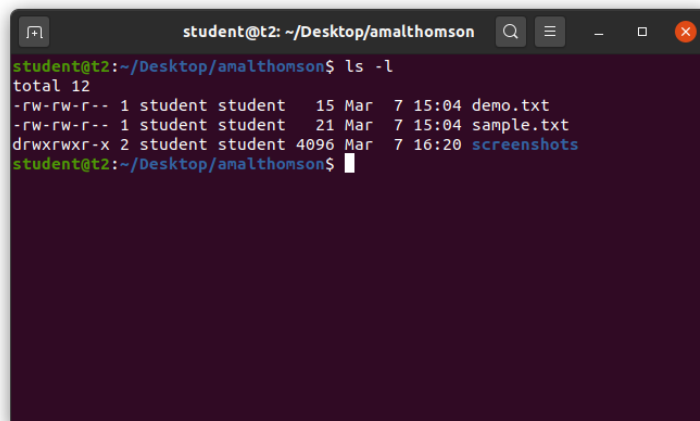
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -R
.:
demo.txt  sample.txt  screenshots

./screenshots:
ls.png  pwd.png
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ ls -l`

Output:

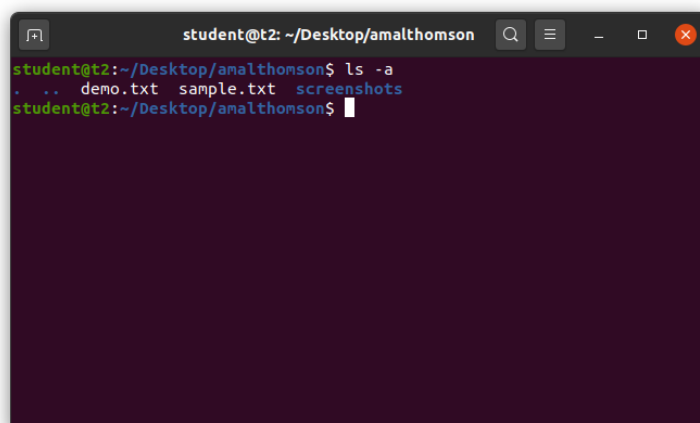


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -l
total 12
-rw-rw-r-- 1 student student 15 Mar 7 15:04 demo.txt
-rw-rw-r-- 1 student student 21 Mar 7 15:04 sample.txt
drwxrwxr-x 2 student student 4096 Mar 7 16:20 screenshots
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ ls -a`

Output:

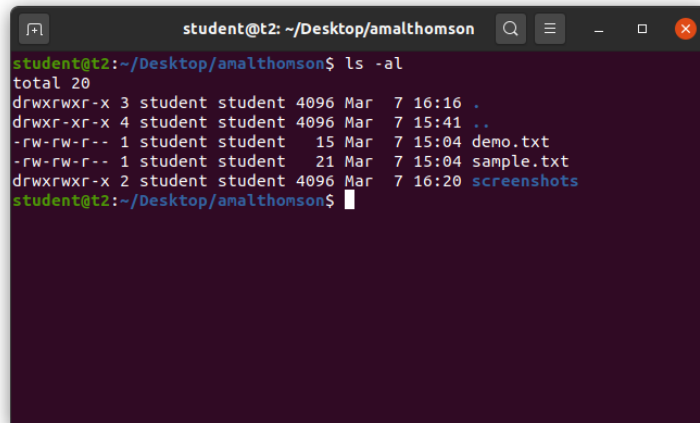


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -a
.  ..  demo.txt  sample.txt  screenshots
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ ls -al`

Output:

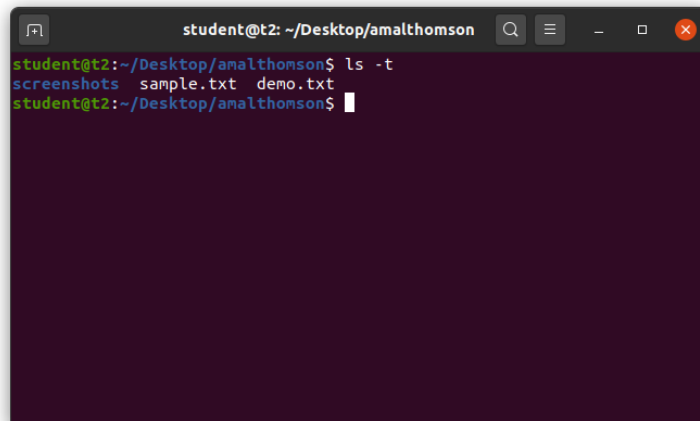


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -al
total 20
drwxrwxr-x 3 student student 4096 Mar  7 16:16 .
drwxr-xr-x 4 student student 4096 Mar  7 15:41 ..
-rw-rw-r-- 1 student student  15 Mar  7 15:04 demo.txt
-rw-rw-r-- 1 student student  21 Mar  7 15:04 sample.txt
drwxrwxr-x 2 student student 4096 Mar  7 16:20 screenshots
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ ls -t`

Output:

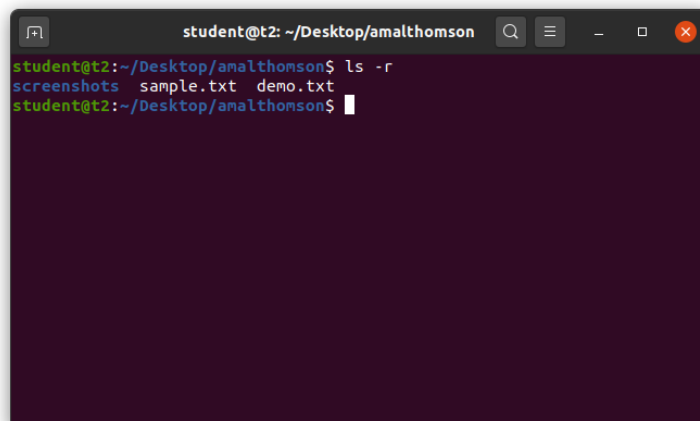


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -t
screenshots sample.txt demo.txt
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ ls -r`

Output:

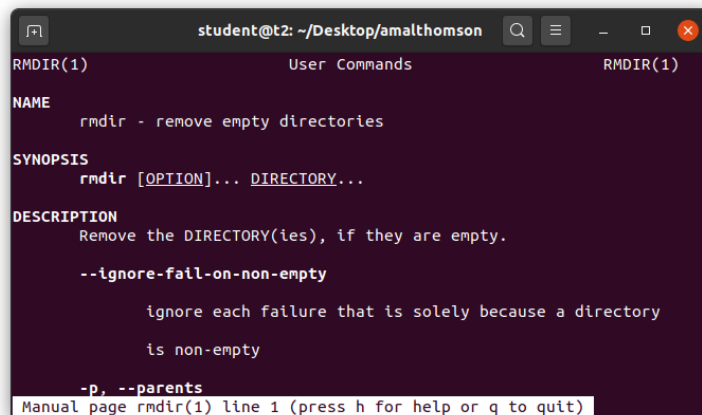


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls -r
screenshots sample.txt demo.txt
student@t2:~/Desktop/amalthomson$
```

3. **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:



```
student@t2: ~/Desktop/amalthomson
RMDIR(1)                                User Commands                                RMDIR(1)

NAME
  rmdir - remove empty directories

SYNOPSIS
  rmdir [OPTION]... DIRECTORY...

DESCRIPTION
  Remove the DIRECTORY(ies), if they are empty.

  --ignore-fail-on-non-empty
                        ignore each failure that is solely because a directory
                        is non-empty

  -p, --parents
Manual page rmdir(1) line 1 (press h for help or q to quit)
```

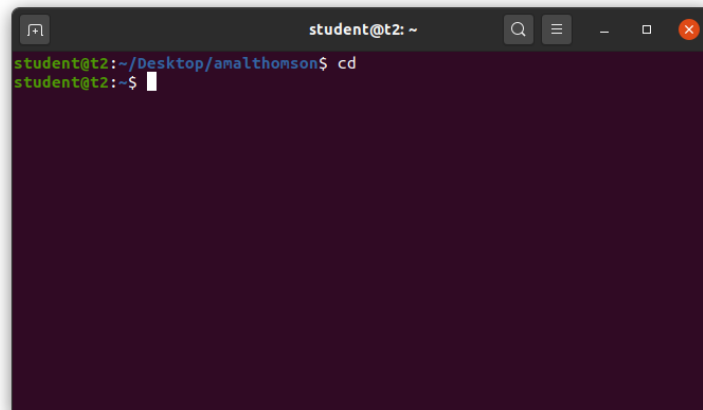
4. **cd** – used to navigate through directory.

Options of cd commands:

- a) **cd** – used to switch to home directory.

Syntax: `$ cd`

Output:

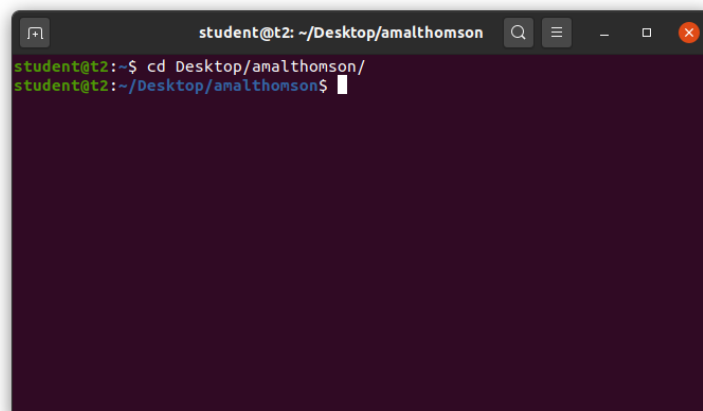


```
student@t2: ~
student@t2: ~/Desktop/amalthomson$ cd
student@t2: ~$
```

- b) **cd <path>** - used to change to a particular path or directory

Syntax: `$ cd <directory_path>`

Output:

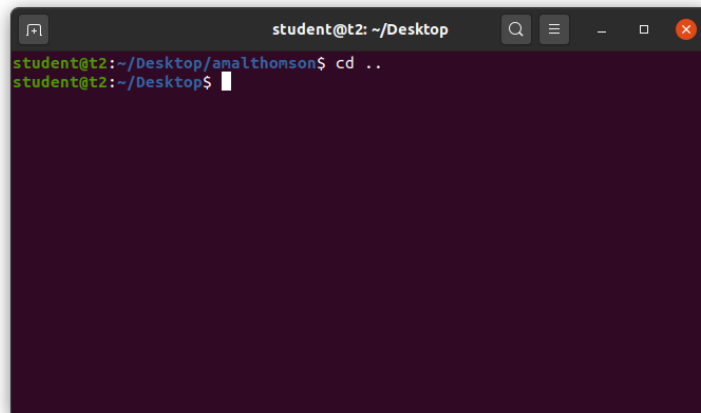


```
student@t2: ~/Desktop/amalthomson
student@t2: ~$ cd Desktop/amalthomson/
student@t2: ~/Desktop/amalthomson$
```

- c) **cd ..** – used to switch back to previous directory or one directory back from the current directory

Syntax: `$ cd ..`

Output:

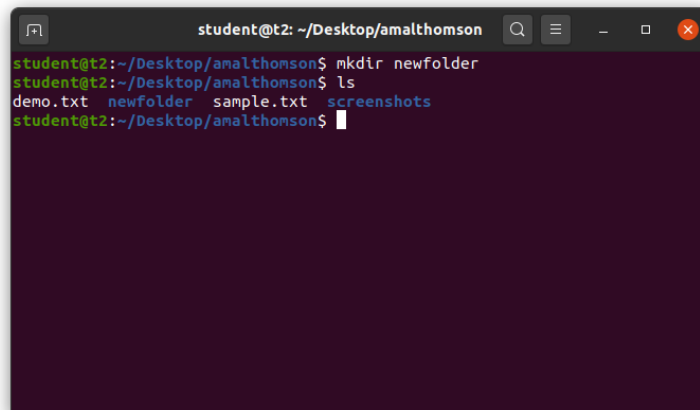


```
student@t2: ~/Desktop
student@t2:~/Desktop/amalthomson$ cd ..
student@t2:~/Desktop$
```

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

Syntax: `$ mkdir <directory_name>`

Output:

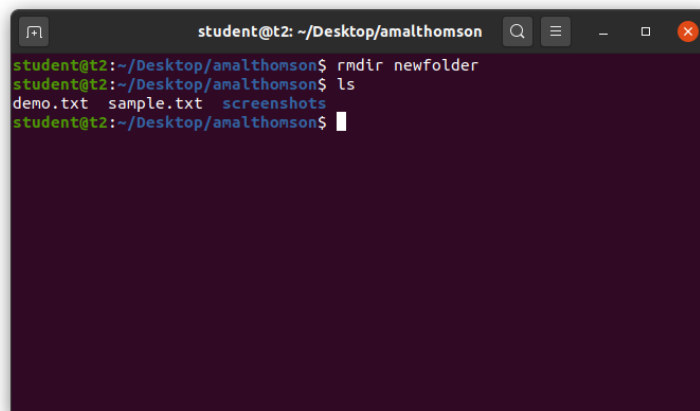


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ mkdir newfolder
student@t2:~/Desktop/amalthomson$ ls
demo.txt  newfolder  sample.txt  screenshots
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ rmdir <directory_name>`

Output:

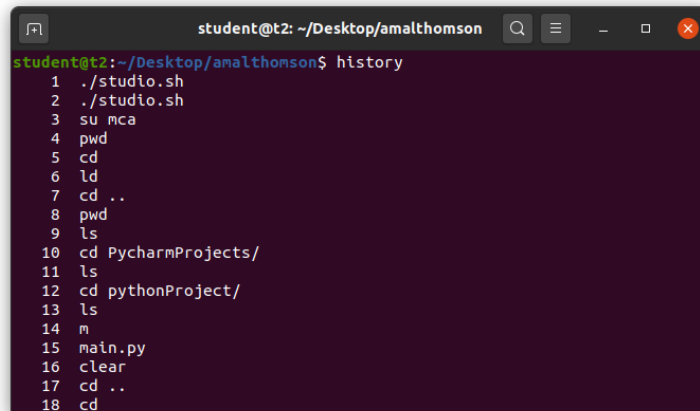


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ rmdir newfolder
student@t2:~/Desktop/amalthomson$ ls
demo.txt  sample.txt  screenshots
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ history

Output;

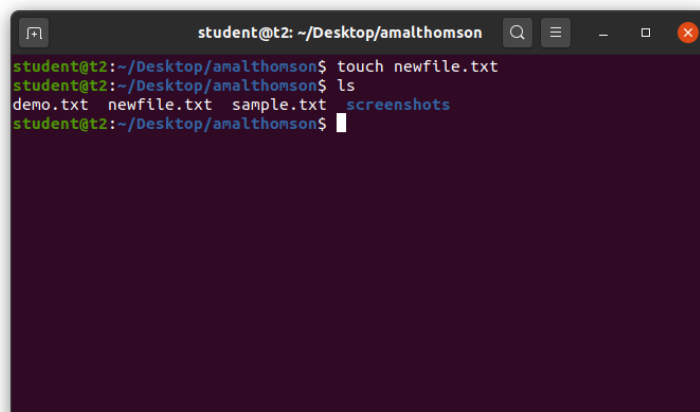


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ history
1  ./studio.sh
2  ./studio.sh
3  su mca
4  pwd
5  cd
6  ld
7  cd ..
8  pwd
9  ls
10 cd PycharmProjects/
11 ls
12 cd pythonProject/
13 ls
14 m
15 main.py
16 clear
17 cd ..
18 cd
```

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

Syntax: \$ touch <filename>

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ touch newfile.txt
student@t2:~/Desktop/amalthomson$ ls
demo.txt  newfile.txt  sample.txt  screenshots
student@t2:~/Desktop/amalthomson$
```

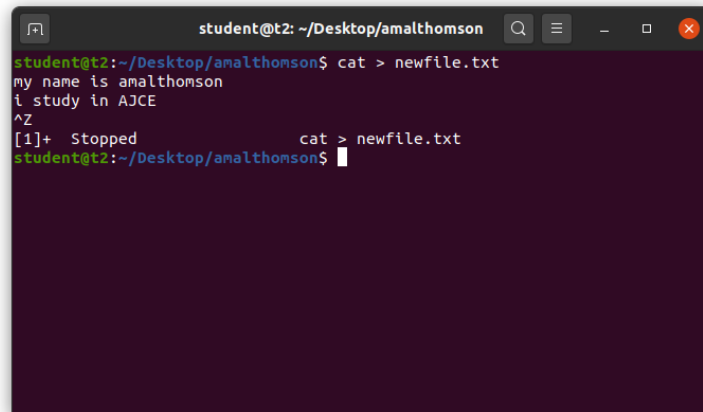
9. **cat** – used to create a new blank file and also to add contents to the file.

Options of cat commands:

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

Syntax: \$ cat > <filename>

Output:

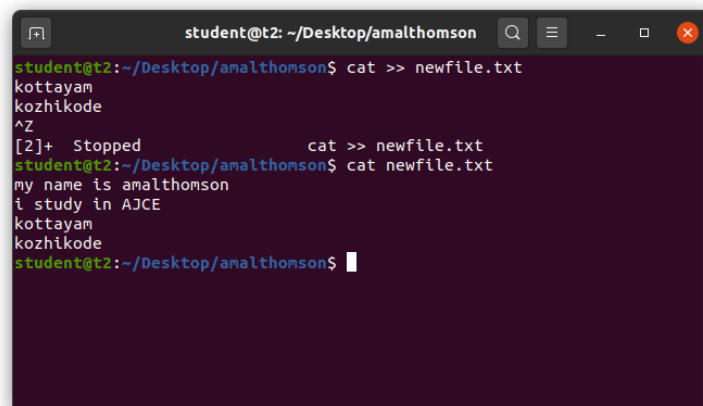


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat > newfile.txt
my name is amalthomson
i study in AJCE
^Z
[1]+  Stopped                  cat > newfile.txt
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ cat >> <filename>`

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat >> newfile.txt
kottayam
kozhikode
^Z
[2]+  Stopped                  cat >> newfile.txt
student@t2:~/Desktop/amalthomson$ cat newfile.txt
my name is amalthomson
i study in AJCE
kottayam
kzhikode
student@t2:~/Desktop/amalthomson$
```

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

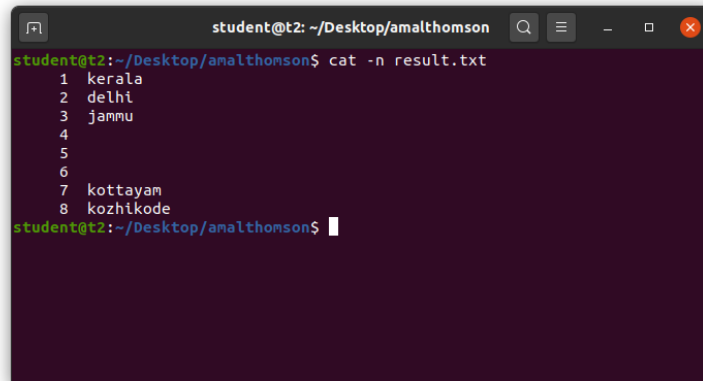
Syntax: `$ cat <filename> <filename> > <filename>`

Output:

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

Syntax: `$ cat -n <filename>`

Output:

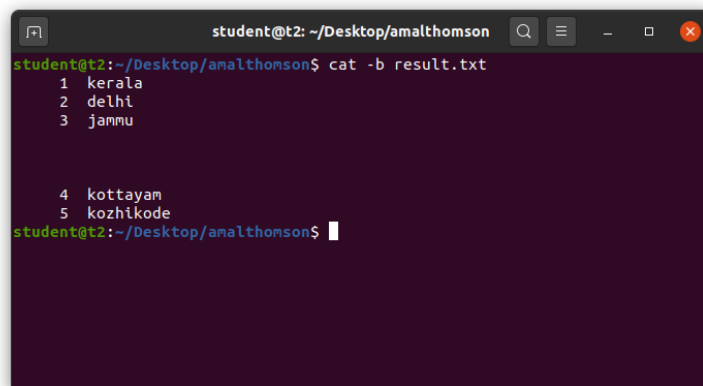


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat -n result.txt
1 kerala
2 delhi
3 jammu
4
5
6
7 kottayam
8 kozhikode
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ cat -b <filename>`

Output:



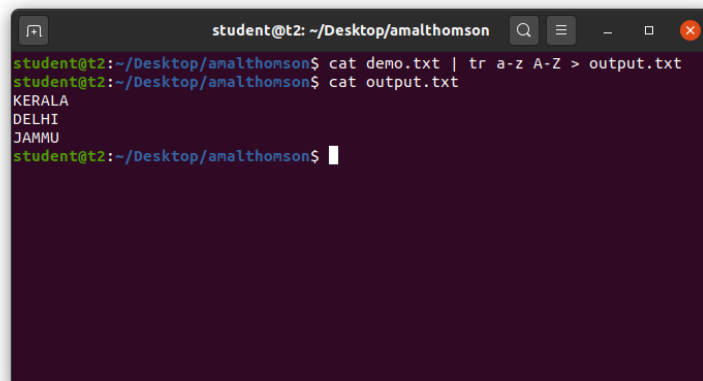
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat -b result.txt
1 kerala
2 delhi
3 jammu

4 kottayam
5 kozhikode
student@t2:~/Desktop/amalthomson$
```

- f) **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:

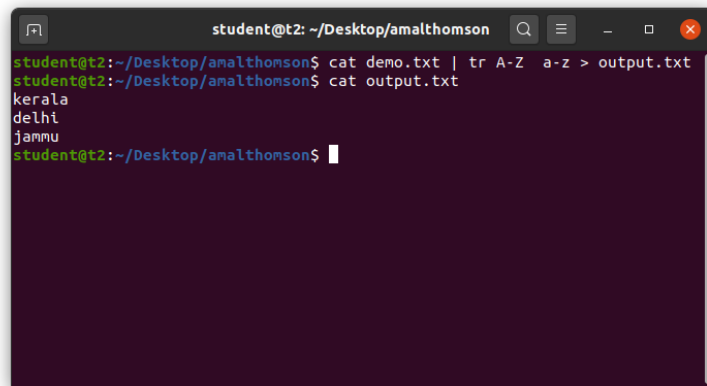


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat demo.txt | tr a-z A-Z > output.txt
student@t2:~/Desktop/amalthomson$ cat output.txt
KERALA
DELHI
JAMMU
student@t2:~/Desktop/amalthomson$
```

- a) **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:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cat demo.txt | tr A-Z a-z > output.txt
student@t2:~/Desktop/amalthomson$ cat output.txt
kerala
delhi
jammu
student@t2:~/Desktop/amalthomson$
```

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

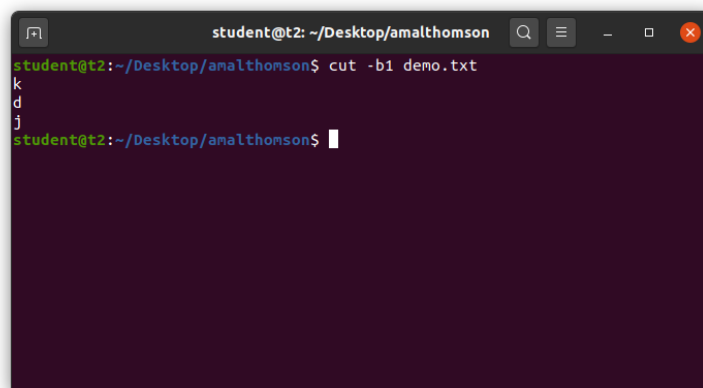
1. **cut** – to cut the contents of the file.

Options of cut command:

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

Syntax: `$ cut -b1 <filename>`

Output:

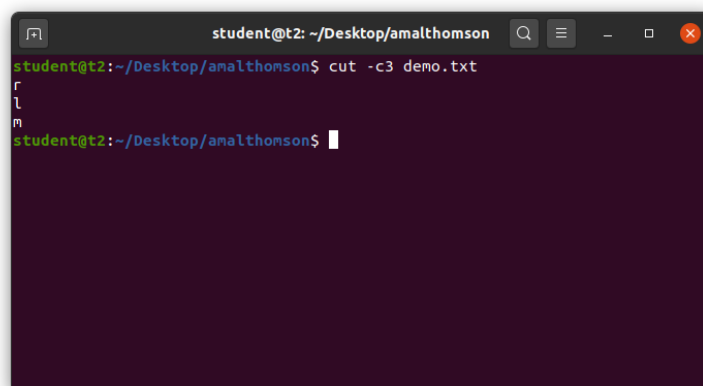


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cut -b1 demo.txt
k
d
j
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ cut -c3 <filename>`

Output:

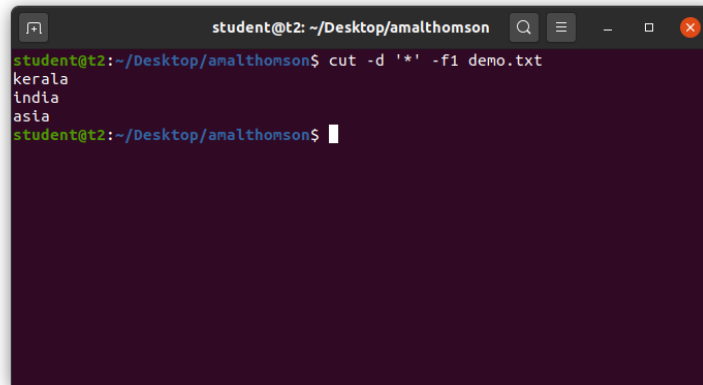


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cut -c3 demo.txt
r
l
m
student@t2:~/Desktop/amalthomson$
```

- c) **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:

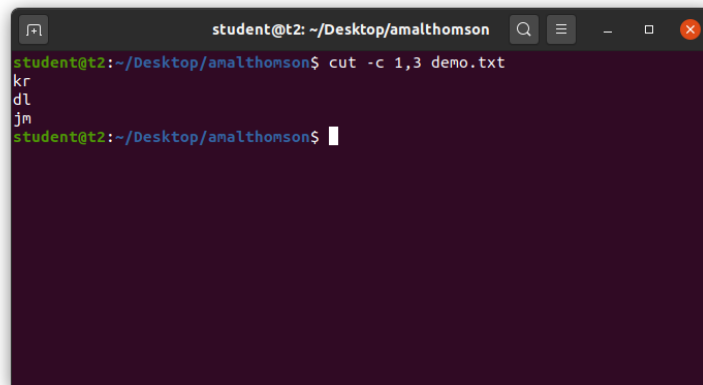


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cut -d '*' -f1 demo.txt
kerala
india
asia
student@t2:~/Desktop/amalthomson$
```

- d) **cut -c** – to cut the characters from a specified position in a file.

Syntax: `$ cut -c [1,3] <filename>`

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cut -c 1,3 demo.txt
kr
dl
jm
student@t2:~/Desktop/amalthomson$
```

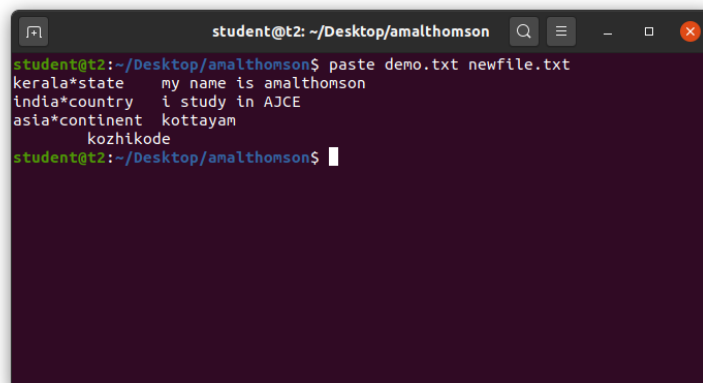
2. **paste** – to paste the content of a file to another.

Options of paste command

- a) **paste <filename> <filename>** – to paste the contents in file1 to file2.

Syntax: `$ paste <filename> <filename>`

Output:

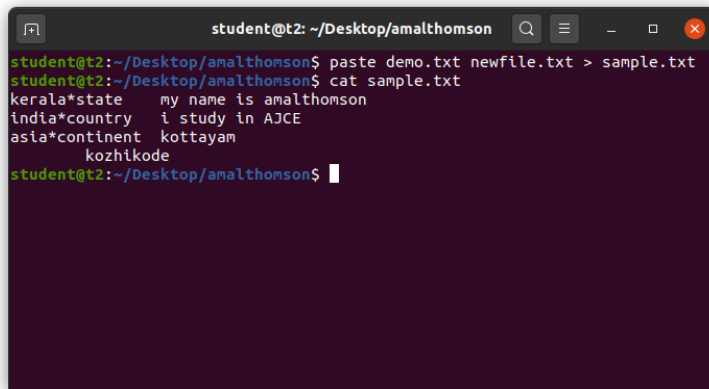


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ paste demo.txt newfile.txt
kerala*state    my name is amalthomson
india*country   i study in AJCE
asia*continent  kottayam
                kozhikode
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ paste <filename> <filename> > <filename>`

Output:

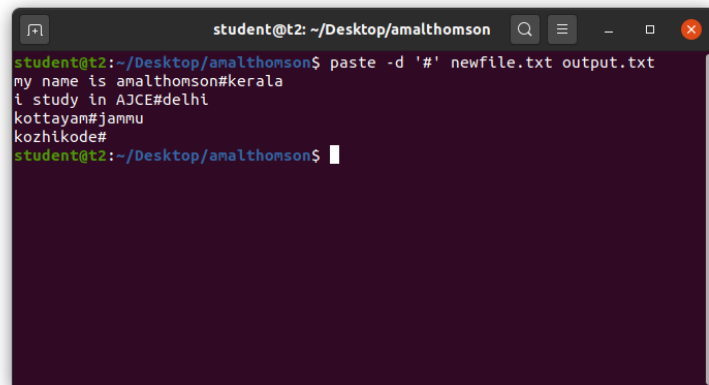


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ paste demo.txt newfile.txt > sample.txt
student@t2:~/Desktop/amalthomson$ cat sample.txt
kerala*state      my name is amalthomson
india*country     i study in AJCE
asia*continent    kottayam
                  kozhikode
student@t2:~/Desktop/amalthomson$
```

- c) **paste -d '#' <filename> <filename>** – to paste # and join the contents of a file with another file.

Syntax: `$ paste -d '#' <filename> <filename>`

Output:

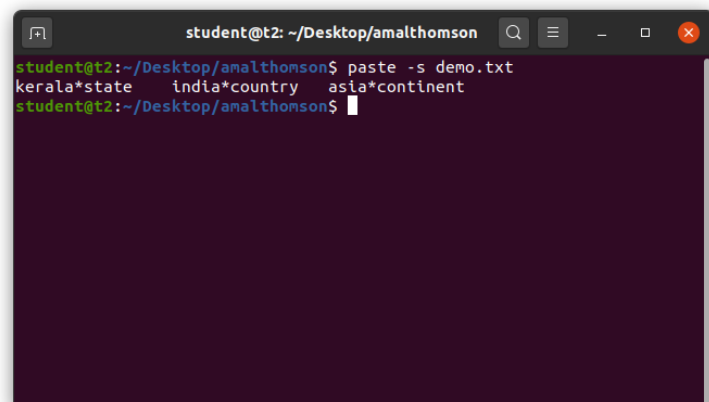


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ paste -d '#' newfile.txt output.txt
my name is amalthomson#kerala
i study in AJCE#delhi
kottayam#jammu
kzhikode#
student@t2:~/Desktop/amalthomson$
```

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

Syntax: `$ paste -s <filename>`

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ paste -s demo.txt
kerala*state      india*country    asia*continent
student@t2:~/Desktop/amalthomson$
```

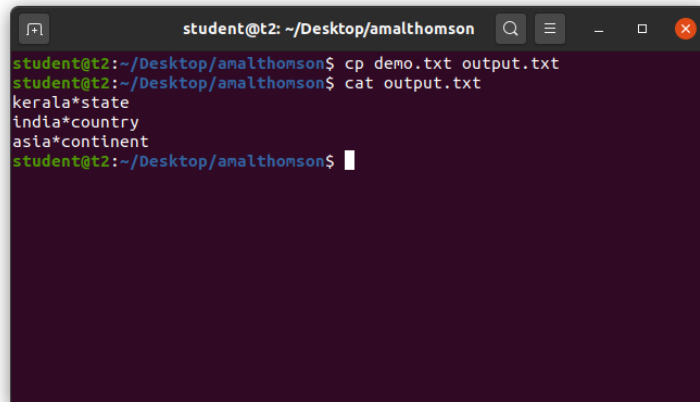
3. **cp** – to copy the contents of a file.

Options of cp command:

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

Syntax: `$ cp <filename> <filename>`

Output:

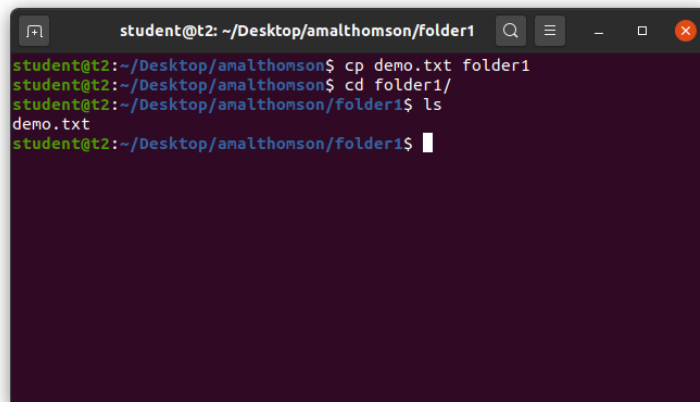


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ cp demo.txt output.txt
student@t2:~/Desktop/amalthomson$ cat output.txt
kerala*state
india*country
asia*continent
student@t2:~/Desktop/amalthomson$
```

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

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

Output:

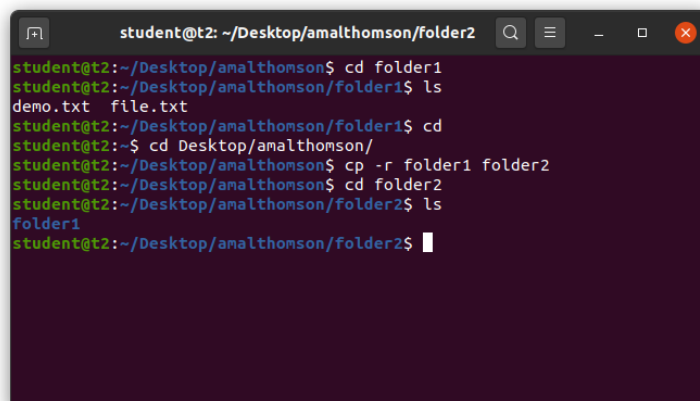


```
student@t2: ~/Desktop/amalthomson/folder1
student@t2:~/Desktop/amalthomson$ cp demo.txt folder1
student@t2:~/Desktop/amalthomson$ cd folder1/
student@t2:~/Desktop/amalthomson/folder1$ ls
demo.txt
student@t2:~/Desktop/amalthomson/folder1$
```

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

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

Output:



```
student@t2: ~/Desktop/amalthomson/folder2
student@t2:~/Desktop/amalthomson$ cd folder1
student@t2:~/Desktop/amalthomson/folder1$ ls
demo.txt  file.txt
student@t2:~/Desktop/amalthomson/folder1$ cd
student@t2:~$ cd Desktop/amalthomson/
student@t2:~/Desktop/amalthomson$ cp -r folder1 folder2
student@t2:~/Desktop/amalthomson$ cd folder2
student@t2:~/Desktop/amalthomson/folder2$ ls
folder1
student@t2:~/Desktop/amalthomson/folder2$
```

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

Options of read command

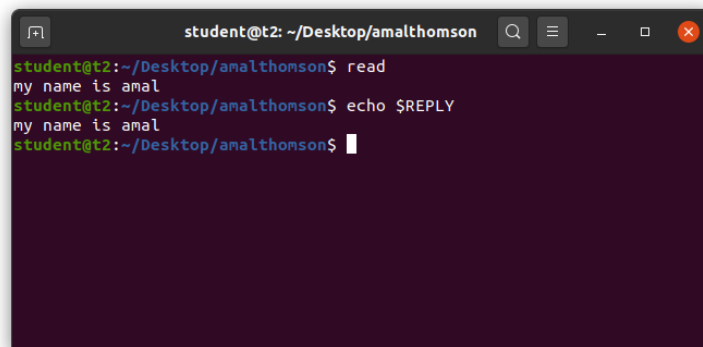
- a) **read** – read contents of a line into variable.

Syntax: \$ read

My name is amal

\$ echo \$REPLY

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ read
my name is amal
student@t2:~/Desktop/amalthomson$ echo $REPLY
my name is amal
student@t2:~/Desktop/amalthomson$
```

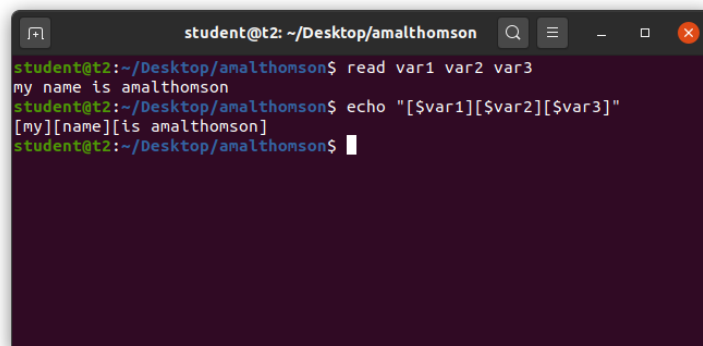
- b) **read <variable_name>** – read contents of a line to a particular variables.

Syntax: \$ read var1 var2 var3

My name is amalthomson

\$ echo "[var1][var2][var3]"

Output:



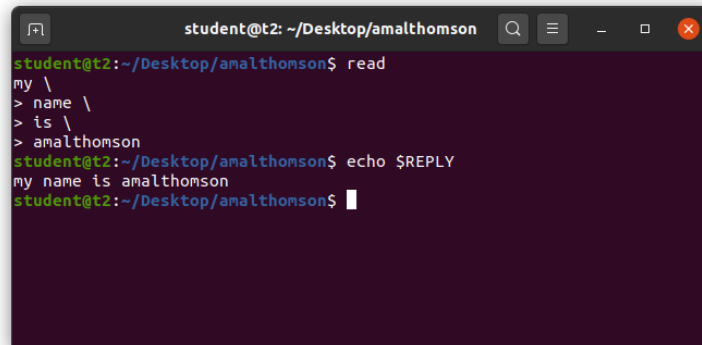
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ read var1 var2 var3
my name is amalthomson
student@t2:~/Desktop/amalthomson$ echo "[$var1][$var2][$var3]"
[my][name][is amalthomson]
student@t2:~/Desktop/amalthomson$
```

- c) **read** – read from multiple lines

Syntax:

```
$ read  
my \  
name \  
is \  
amalthomson  
$ echo $REPLY
```

Outout:

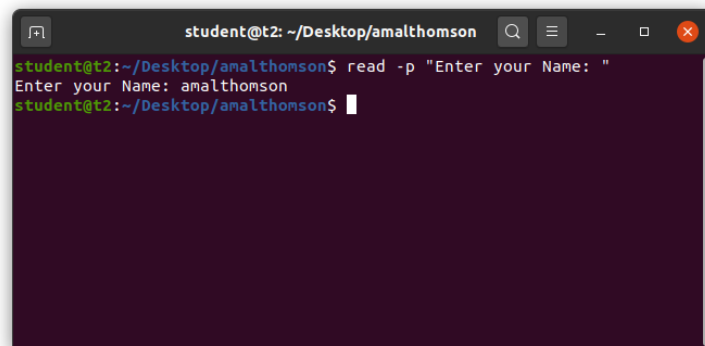


```
student@t2: ~/Desktop/amalthomson  
student@t2:~/Desktop/amalthomson$ read  
my \  
> name \  
> is \  
> amalthomson  
student@t2:~/Desktop/amalthomson$ echo $REPLY  
my name is amalthomson  
student@t2:~/Desktop/amalthomson$
```

d) **read -p** – read with prompt message

Syntax: \$ read -p “Enter your name”

Output:




```
student@t2: ~/Desktop/amalthomson  
student@t2:~/Desktop/amalthomson$ read -p "Enter your Name: "  
Enter your Name: amalthomson  
student@t2:~/Desktop/amalthomson$
```

e) **read -n** – read with limit of characters can be read

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

Output:

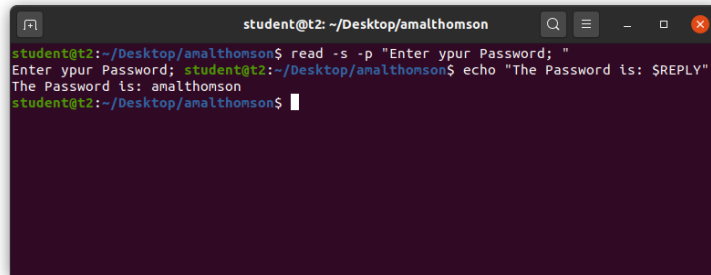


```
student@t2: ~/Desktop/amalthomson  
student@t2:~/Desktop/amalthomson$ read -n 6 -p "Enter Six Characters Only; "  
Enter Six Characters Only; amal  
student@t2:~/Desktop/amalthomson$ read -n 6 -p "Enter Six Characters Only; "  
student@t2:~/Desktop/amalthomson$
```

f) **read -s** – read lines securely without displaying the data entered

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

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ read -s -p "Enter your Password: "
Enter your Password; student@t2:~/Desktop/amalthomson$ echo "The Password is: $REPLY"
The Password is: amalthomson
student@t2:~/Desktop/amalthomson$
```

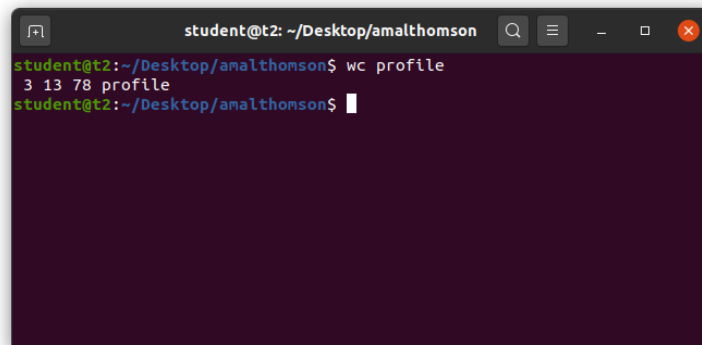
2. **wc** – word count

Options of **wc** commands

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

Syntax: \$ wc profile.txt

Output:

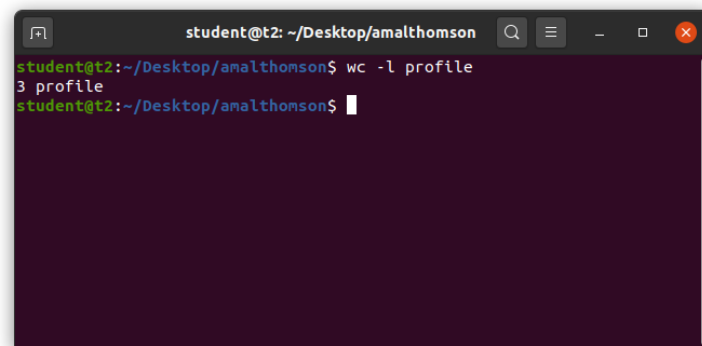


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc profile
3 13 78 profile
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ wc -l profile.txt

Output:

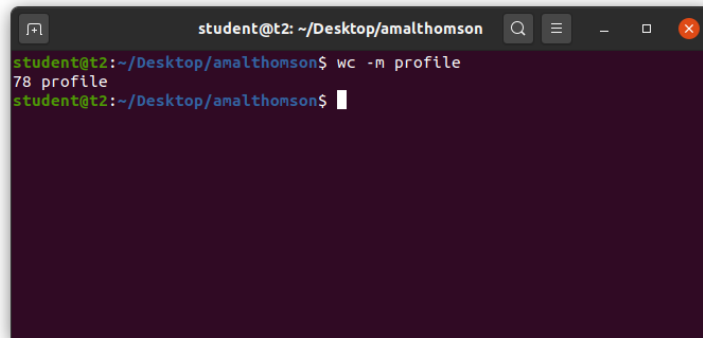


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc -l profile
3 profile
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ wc -m profile.txt

Output:

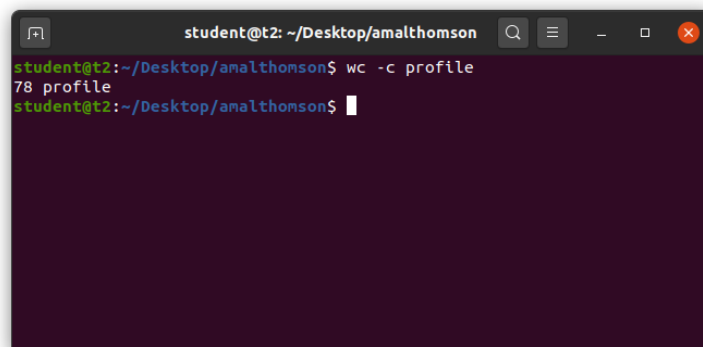


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc -m profile
78 profile
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ wc -c profile.txt

Output:

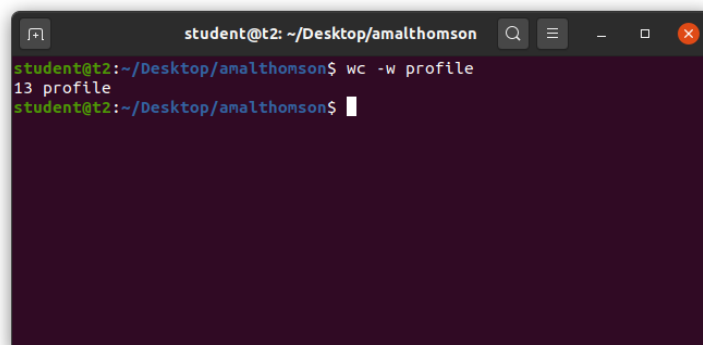


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc -c profile
78 profile
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ wc -w profile.txt

Output:

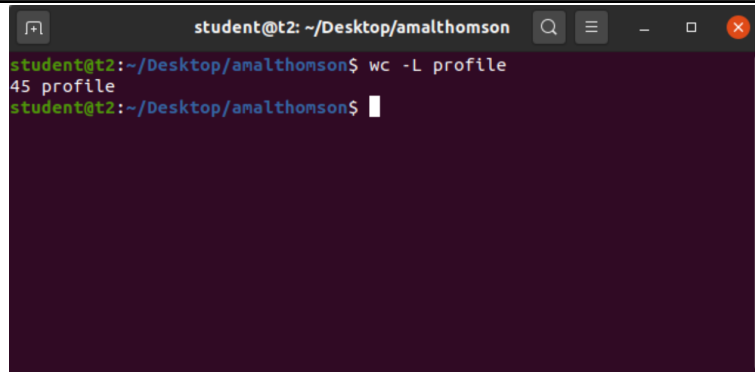


```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc -w profile
13 profile
student@t2:~/Desktop/amalthomson$
```

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

Syntax: \$ wc -L profile.txt

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ wc -L profile
45 profile
student@t2:~/Desktop/amalthomson$
```

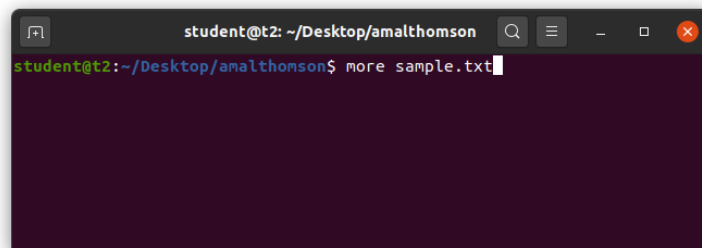
3. **more** – is similar to get to display the contents, the only difference is that in case of longer text or content get command output will scroll off your screen while more command display the output only screen full at a time.

Options of more command

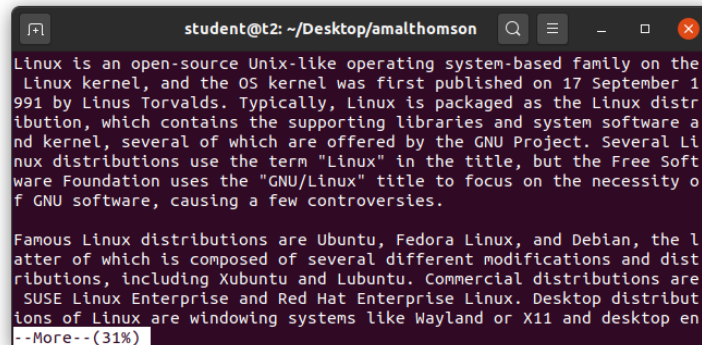
- a) **more <filename>** – display contents of a file

Syntax: \$ more sample.txt

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ more sample.txt
```



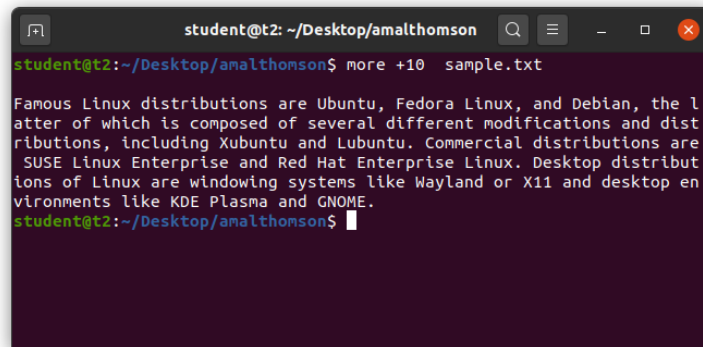
```
student@t2: ~/Desktop/amalthomson
Linux is an open-source Unix-like operating system-based family on the
Linux kernel, and the OS kernel was first published on 17 September 1
991 by Linus Torvalds. Typically, Linux is packaged as the Linux distr
ibution, which contains the supporting libraries and system software a
nd kernel, several of which are offered by the GNU Project. Several Li
nux distributions use the term "Linux" in the title, but the Free Soft
ware Foundation uses the "GNU/Linux" title to focus on the necessity o
f GNU software, causing a few controversies.

Famous Linux distributions are Ubuntu, Fedora Linux, and Debian, the l
atter of which is composed of several different modifications and dist
ributions, including Xubuntu and Lubuntu. Commercial distributions are
SUSE Linux Enterprise and Red Hat Enterprise Linux. Desktop distribut
ions of Linux are windowing systems like Wayland or X11 and desktop en
--More-- (31%)
```

- b) **more +20 <filename>** – display contents of a file

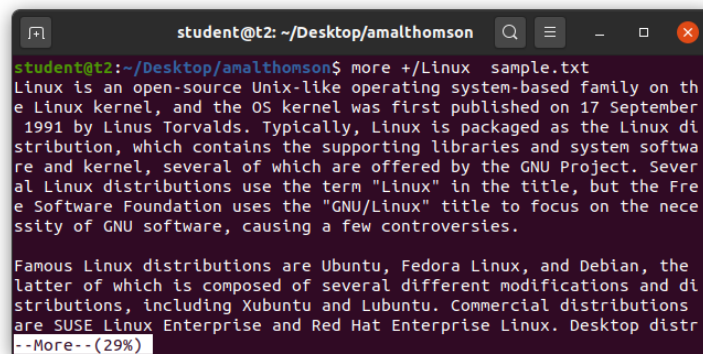
Syntax: \$ more +10 sample.txt

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ more +10 sample.txt
Famous Linux distributions are Ubuntu, Fedora Linux, and Debian, the latter of which is composed of several different modifications and distributions, including Xubuntu and Lubuntu. Commercial distributions are SUSE Linux Enterprise and Red Hat Enterprise Linux. Desktop distributions of Linux are windowing systems like Wayland or X11 and desktop environments like KDE Plasma and GNOME.
student@t2:~/Desktop/amalthomson$
```

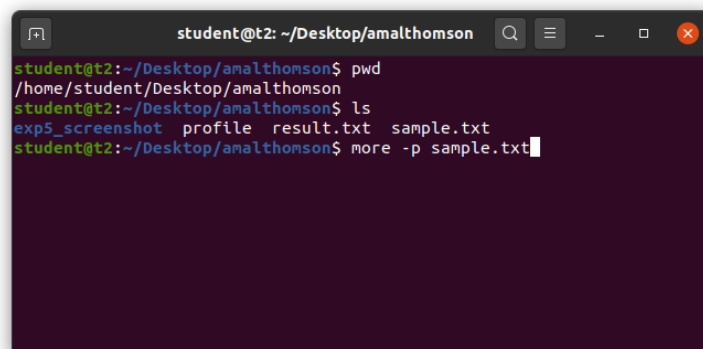
- c) **more +/pattern <filename>** – to search is train inside your document, you can view all the instances by navigating through the result
Syntax: `$ more +/Linux sample.txt`
Output:



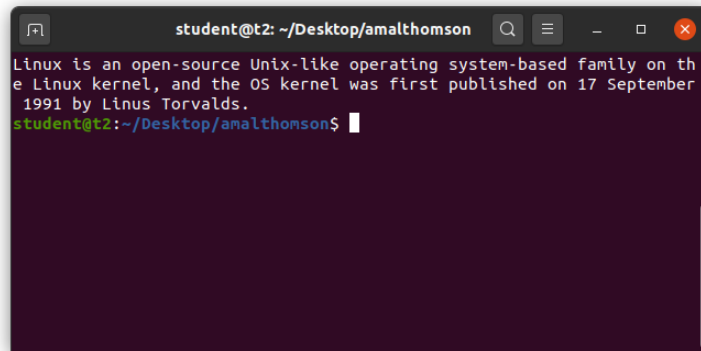
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ more +/Linux sample.txt
Linux is an open-source Unix-like operating system-based family on the Linux kernel, and the OS kernel was first published on 17 September 1991 by Linus Torvalds. Typically, Linux is packaged as the Linux distribution, which contains the supporting libraries and system software and kernel, several of which are offered by the GNU Project. Several Linux distributions use the term "Linux" in the title, but the Free Software Foundation uses the "GNU/Linux" title to focus on the necessity of GNU software, causing a few controversies.

Famous Linux distributions are Ubuntu, Fedora Linux, and Debian, the latter of which is composed of several different modifications and distributions, including Xubuntu and Lubuntu. Commercial distributions are SUSE Linux Enterprise and Red Hat Enterprise Linux. Desktop distributions of Linux are windowing systems like Wayland or X11 and desktop environments like KDE Plasma and GNOME.
--More-- (29%)
```

- d) **more -p <filename>** – to display the contents of a file after clearing the screen
Syntax: `$ more -p sample.txt`
Output:



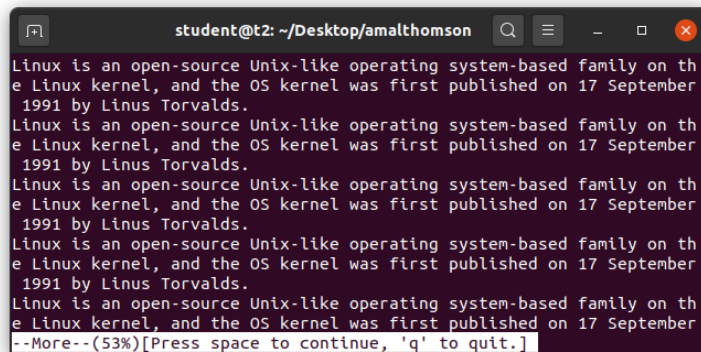
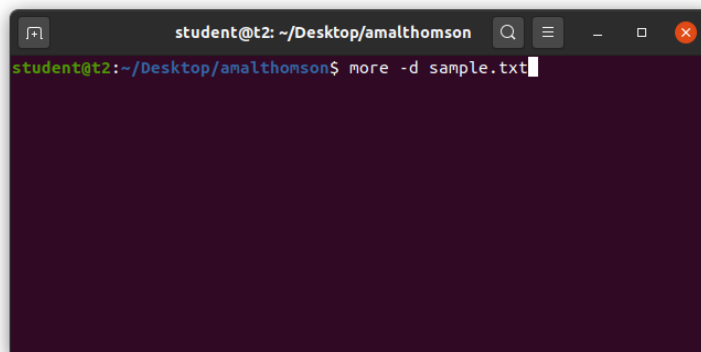
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ pwd
/home/student/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ ls
exp5_screenshot  profile  result.txt  sample.txt
student@t2:~/Desktop/amalthomson$ more -p sample.txt
```



- e) **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.: 7

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.

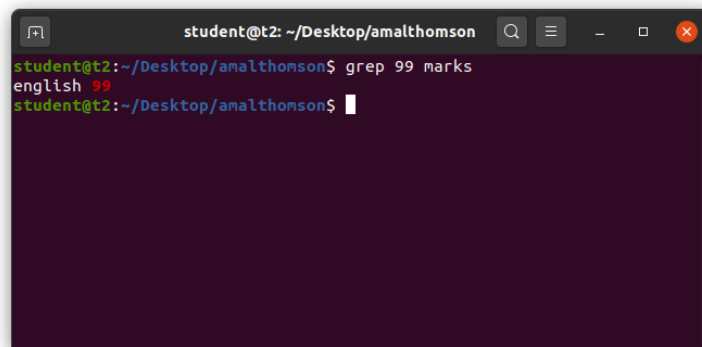
Options of `grep` command

- a) `grep <content><filename>` – search and display a particular content from a file

Syntax:

`$ grep 90 marks`

Output:



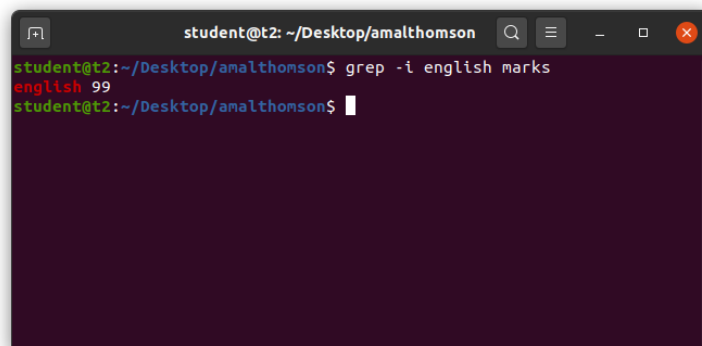
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep 99 marks
english 99
student@t2:~/Desktop/amalthomson$
```

- b) `grep -i <pattern><filename>` – used to search and display a matching pattern, case insensitive

Syntax:

`$ grep -i english marks`

Output:



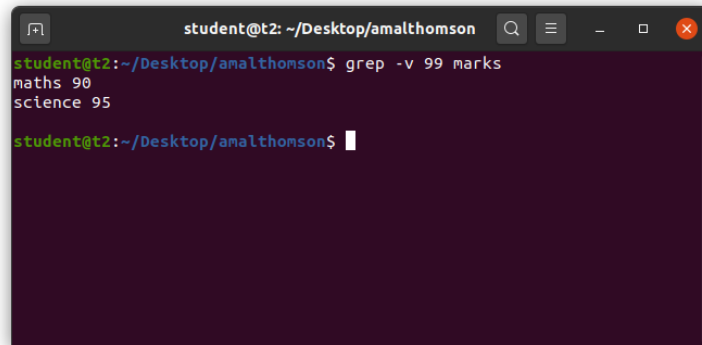
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep -i english marks
english 99
student@t2:~/Desktop/amalthomson$
```

- c) `grep -v <content><filename>` – inverted search and display

Syntax:

\$ grep -v 99 marks

Output:



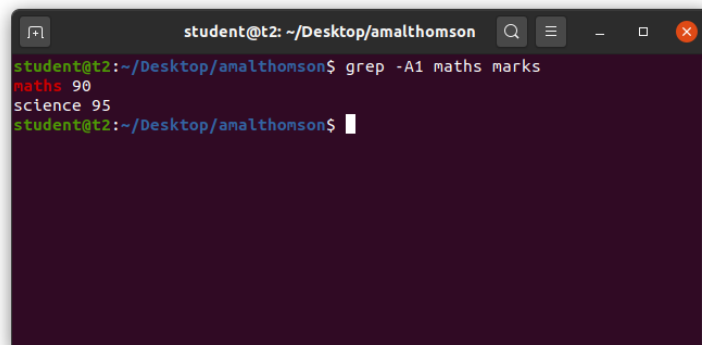
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep -v 99 marks
maths 90
science 95
student@t2:~/Desktop/amalthomson$
```

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

Syntax:

\$ grep -A1 maths marks

Output:



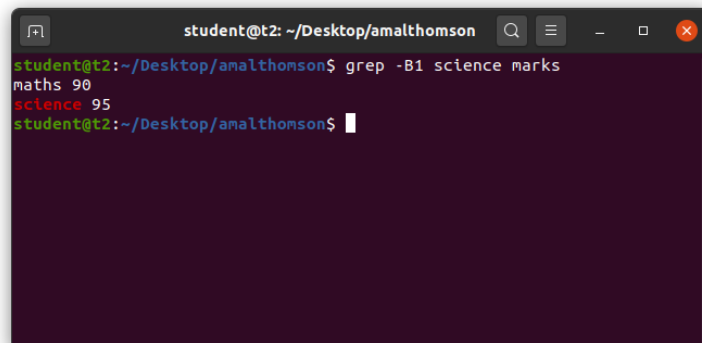
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep -A1 maths marks
maths 90
science 95
student@t2:~/Desktop/amalthomson$
```

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

Syntax:

\$ grep -B1 science marks

Output:



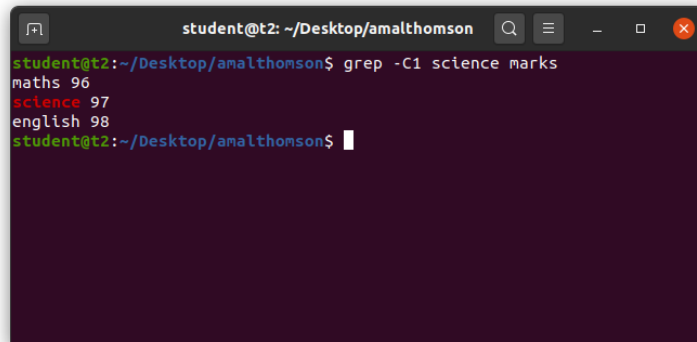
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep -B1 science marks
maths 90
science 95
student@t2:~/Desktop/amalthomson$
```

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

Syntax:

\$ grep -C1 science marks

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ grep -C1 science marks
maths 96
science 97
english 98
student@t2:~/Desktop/amalthomson$
```

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

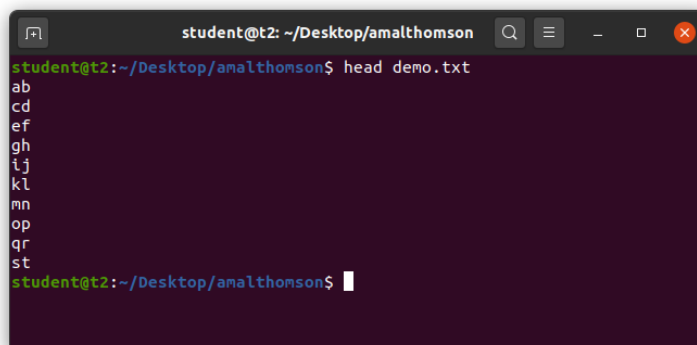
Options of head command.

- a) head <filename> – display top 10 lines of a file

Syntax:

\$ head demo.txt

Output:



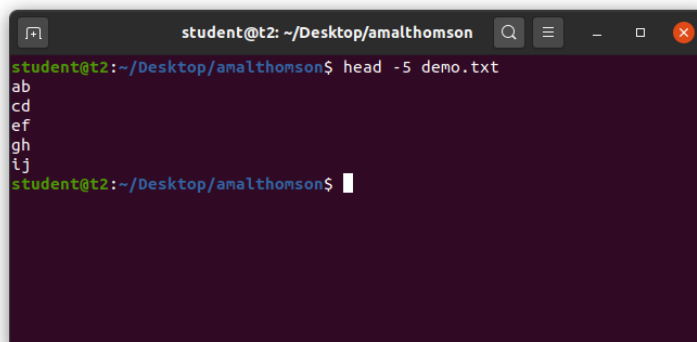
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ head demo.txt
ab
cd
ef
gh
ij
kl
mn
op
qr
st
student@t2:~/Desktop/amalthomson$
```

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

Syntax:

\$ head -5 demo.txt

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ head -5 demo.txt
ab
cd
ef
gh
ij
student@t2:~/Desktop/amalthomson$
```

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

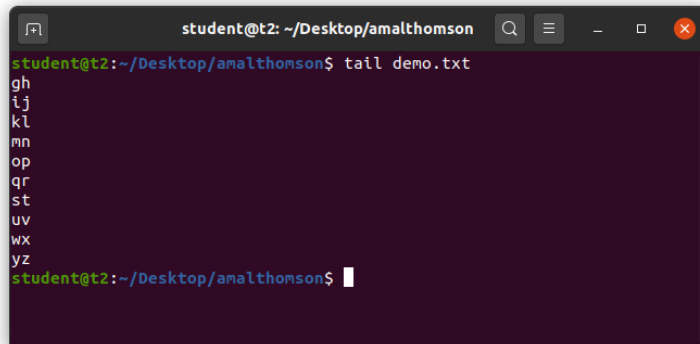
Options of tail command.

- a) `tail <filename>` – display bottom 10 lines of a file

Syntax:

`$ tail demo.txt`

Output:

A terminal window titled 'student@t2: ~/Desktop/amalthomson' showing the command 'tail demo.txt' and its output. The output consists of the last 10 lines of the file 'demo.txt', which are 'gh', 'tj', 'kl', 'mn', 'op', 'qr', 'st', 'uv', 'wx', and 'yz'.

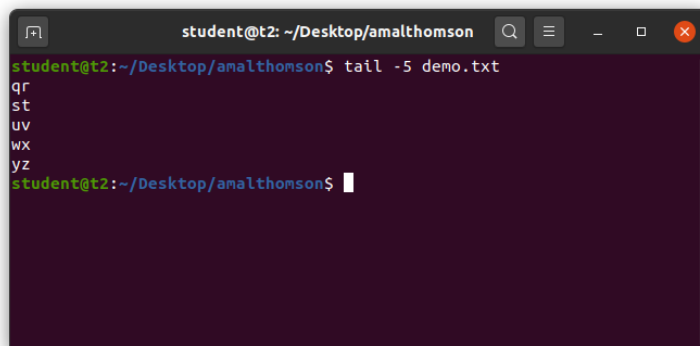
```
student@t2: ~/Desktop/amalthomson$ tail demo.txt
gh
tj
kl
mn
op
qr
st
uv
wx
yz
student@t2: ~/Desktop/amalthomson$
```

- b) `tail -<limit> <filename>` – display bottom number of lines mentioned in the limit of a file

Syntax:

`$ tail -5 demo.txt`

Output:

A terminal window titled 'student@t2: ~/Desktop/amalthomson' showing the command 'tail -5 demo.txt' and its output. The output consists of the last 5 lines of the file 'demo.txt', which are 'qr', 'st', 'uv', 'wx', and 'yz'.

```
student@t2: ~/Desktop/amalthomson$ tail -5 demo.txt
qr
st
uv
wx
yz
student@t2: ~/Desktop/amalthomson$
```

4. `mv` – used to move files or folders.

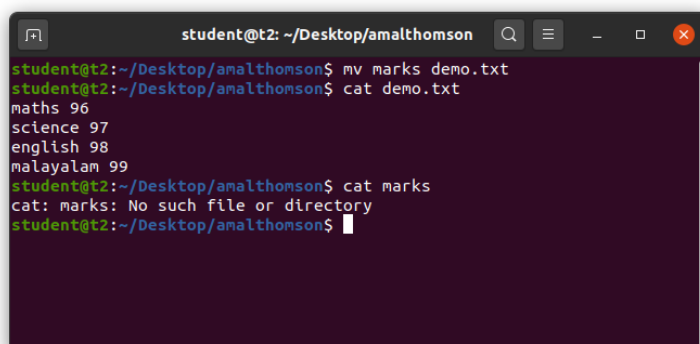
Options of move command

- a) `mv <filename> <filename>` – replaces file2 with file1

Syntax:

`$ mv marks demo.txt`

Output:

A terminal window titled 'student@t2: ~/Desktop/amalthomson' showing the command 'mv marks demo.txt' and its output. The output shows the contents of 'demo.txt' (marks, science, english, malayalam) and then the command 'cat marks' which results in an error message 'cat: marks: No such file or directory' because the file has been renamed to demo.txt.

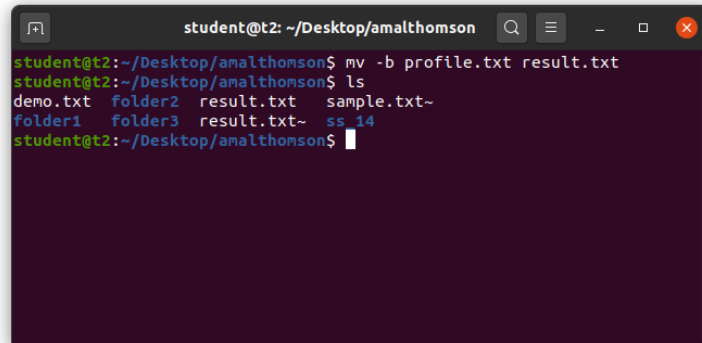
```
student@t2: ~/Desktop/amalthomson$ mv marks demo.txt
student@t2: ~/Desktop/amalthomson$ cat demo.txt
marks 96
science 97
english 98
malayalam 99
student@t2: ~/Desktop/amalthomson$ cat marks
cat: marks: No such file or directory
student@t2: ~/Desktop/amalthomson$
```


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

Syntax:

`$ mv -b profile.txt result.txt`

Output:



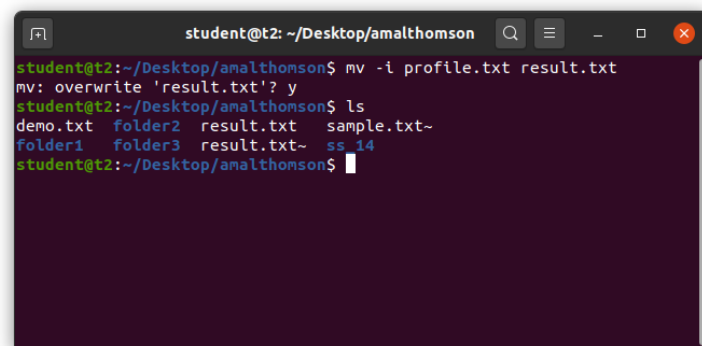
```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ mv -b profile.txt result.txt
student@t2:~/Desktop/amalthomson$ ls
demo.txt  folder2  result.txt  sample.txt~
folder1   folder3  result.txt~ ss_14
student@t2:~/Desktop/amalthomson$
```

- c) `mv -i <filename> <filename>` – displays a prompt message to confirm overwrite.

Syntax:

`$ mv -i profile.txt result.txt`

Output:



```
student@t2: ~/Desktop/amalthomson
student@t2:~/Desktop/amalthomson$ mv -i profile.txt result.txt
mv: overwrite 'result.txt'? y
student@t2:~/Desktop/amalthomson$ ls
demo.txt  folder2  result.txt  sample.txt~
folder1   folder3  result.txt~ ss_14
student@t2:~/Desktop/amalthomson$
```

Result

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