LAB MANUAL

Operating Systems

I. Basic Unix/Linux Commands

Introduction: Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It was initially released by Linus Torvalds on September 17, 1991. It is a free and open-source operating system and the source code can be modified and distributed to anyone commercially or noncommercial under the GNU General Public License. You can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers, etc.
Some of the popular Linux distributions are:

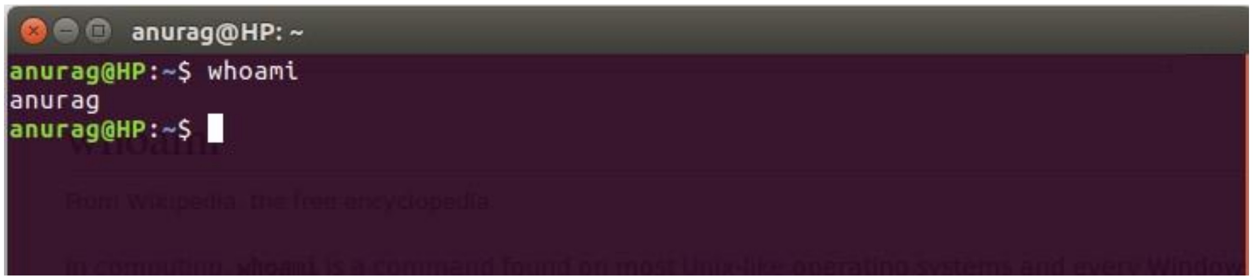
- MX Linux
- Manjaro
- Linux Mint
- Elementary
- Ubuntu
- Debian
- Solus
- Fedora
- openSUSE
- Deepin

File Management becomes easy if you know the right basic command in Linux. Some basic commands of Linux are listed below:

Commands:

1) whoami command

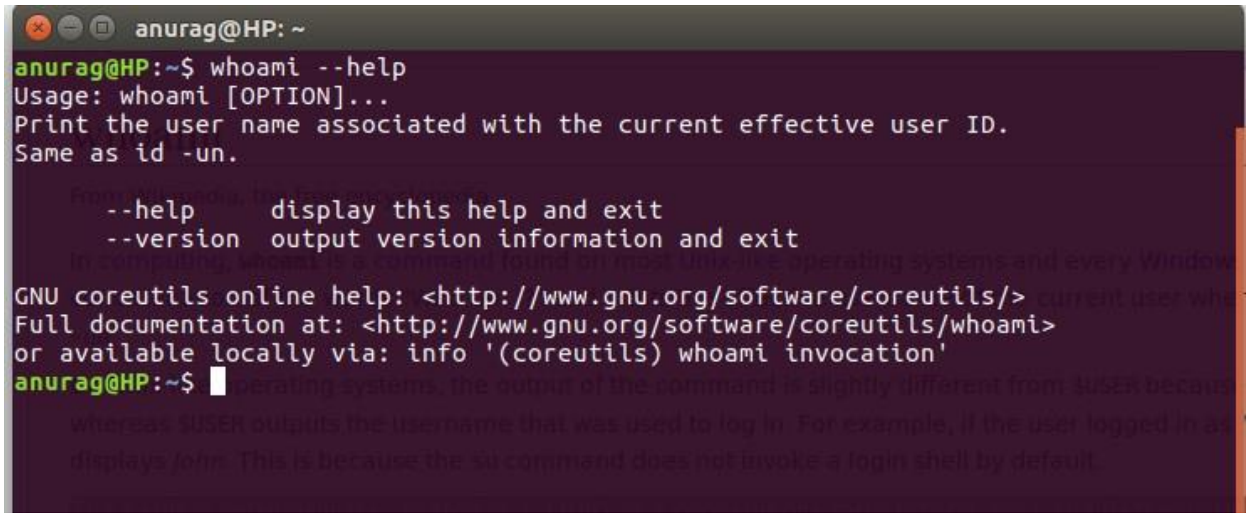
It displays the username of the current user when this command is invoked.



```
anurag@HP:~$ whoami
anurag
anurag@HP:~$
```

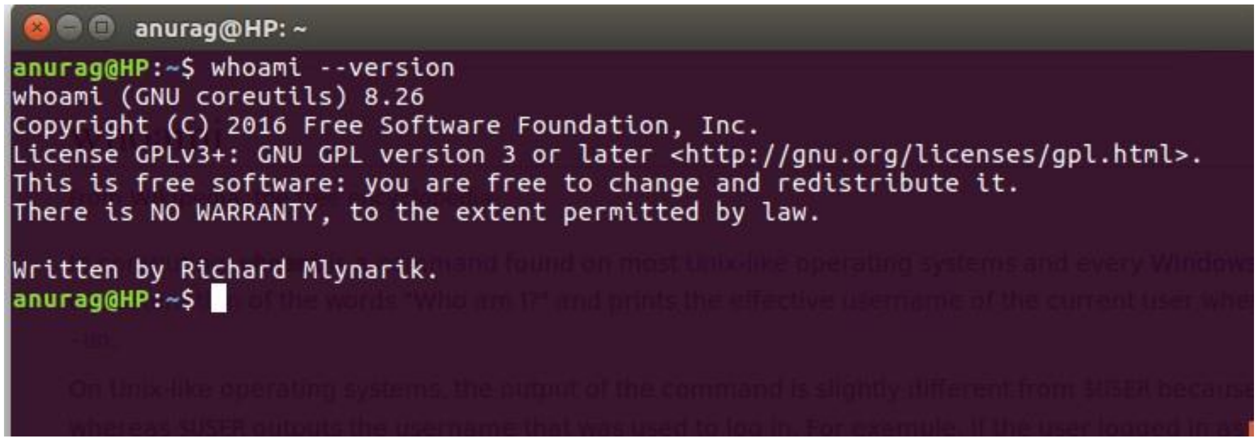
--Help option :-

It gives the help message and exit.
Whoami - - help



```
anurag@HP:~$ whoami --help
Usage: whoami [OPTION]...
Print the user name associated with the current effective user ID.
Same as id -un.

      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/whoami>
or available locally via: info '(coreutils) whoami invocation'
anurag@HP:~$
```

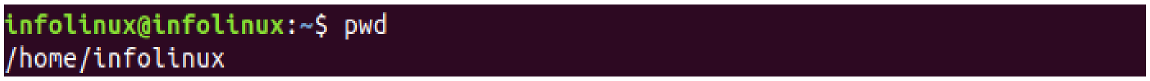
--version Option :-
It gives the version information and exit.
    Whoami --version



```
anurag@HP:~$ whoami --version
whoami (GNU coreutils) 8.26
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Richard Mlynarik.
anurag@HP:~$
```

2) PWD =

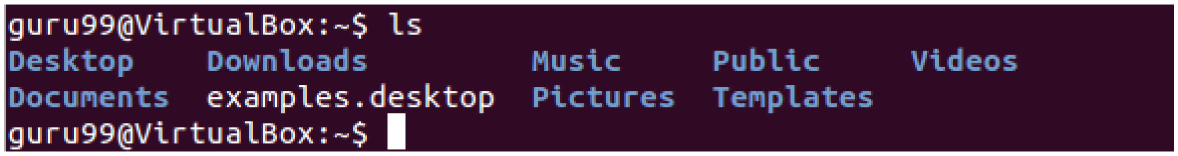
PWD stands for Present working directory. It will simply print out the current working directory.



```
infolinux@infolinux:~$ pwd
/home/infolinux
```

3) ls

It lists the content of a given directory. The peculiarity of this command is that it supports a wide range of arguments.



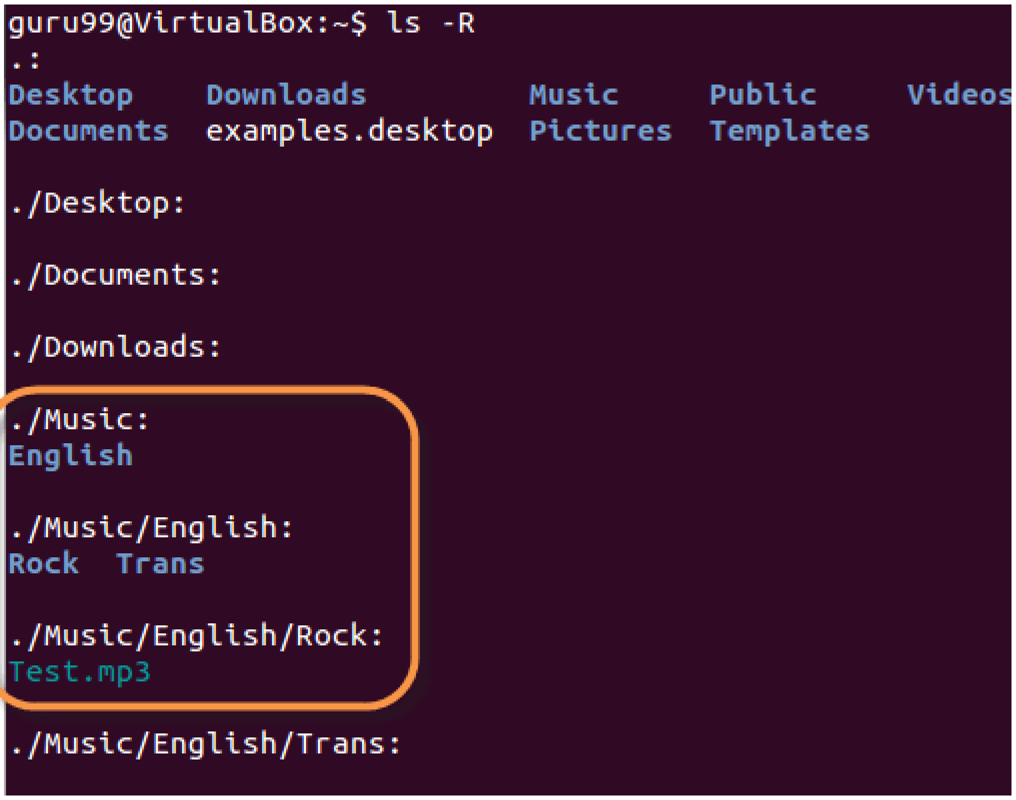
```
guru99@VirtualBox:~$ ls
Desktop     Downloads          Music     Public     Videos
Documents   examples.desktop   Pictures  Templates
guru99@VirtualBox:~$
```

- Directories are denoted in blue colour.
- Files are denoted in white.

4) 'ls-R' command

Suppose, the “Music” folder has some sub-directories and files.

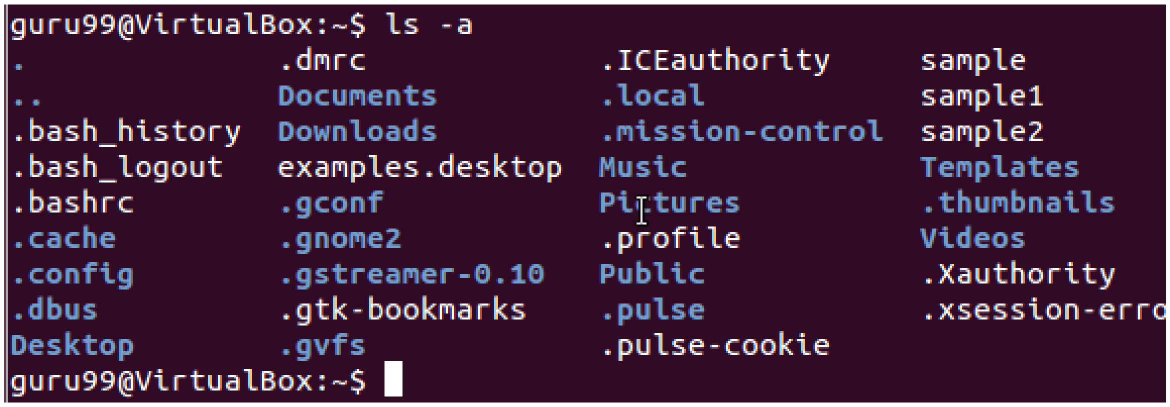
You can use **'ls -R'** to show all the files not only in directories but also subdirectories.



```
guru99@VirtualBox:~$ ls -R
.:
Desktop     Downloads         Music     Public     Videos
Documents   examples.desktop  Pictures  Templates

./Desktop:

./Documents:

./Downloads:

./Music:
English

./Music/English:
Rock   Trans

./Music/English/Rock:
Test.mp3

./Music/English/Trans:
```

5) "ls -a"

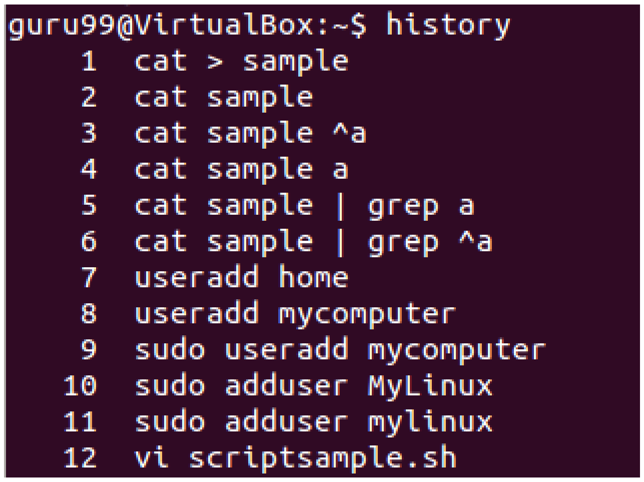
To view hidden files ls-a command used.



```
guru99@VirtualBox:~$ ls -a
.                .dmrc              .ICEauthority     sample
..               Documents          .local            sample1
.bash_history    Downloads          .mission-control  sample2
.bash_logout     examples.desktop   Music             Templates
.bashrc          .gconf             Pictures          .thumbnails
.cache           .gnome2            .profile          Videos
.config          .gstreamer-0.10    Public            .Xauthority
.dbus            .gtk-bookmarks     .pulse            .xsession-erro
Desktop          .gvfs              .pulse-cookie
guru99@VirtualBox:~$
```

6) History Command

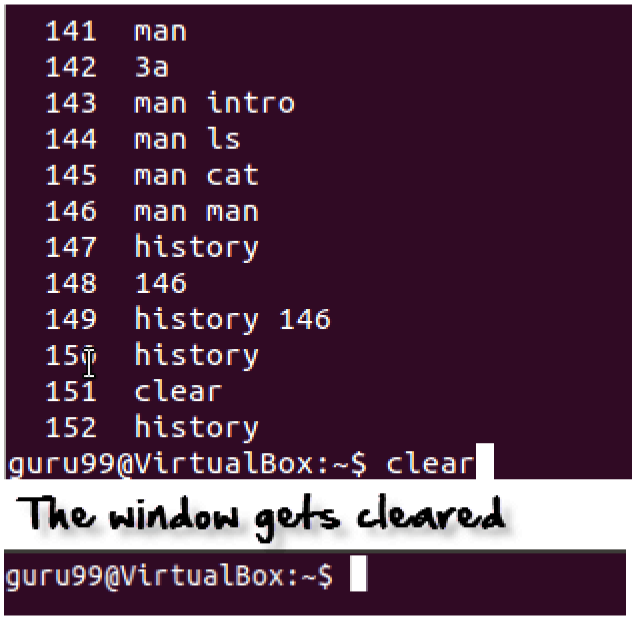
History command shows all the basic commands in Linux that you have used in the past for the current terminal session. This can help you refer to the old commands you have entered and re- used them in your operations again.



```
guru99@VirtualBox:~$ history
    1  cat > sample
    2  cat sample
    3  cat sample ^a
    4  cat sample a
    5  cat sample | grep a
    6  cat sample | grep ^a
    7  useradd home
    8  useradd mycomputer
    9  sudo useradd mycomputer
   10  sudo adduser MyLinux
   11  sudo adduser mylinux
   12  vi scriptsample.sh
```

7) Clear command

This command clears all the clutter on the terminal and gives you a clean window to work on, just like when you launch the terminal.



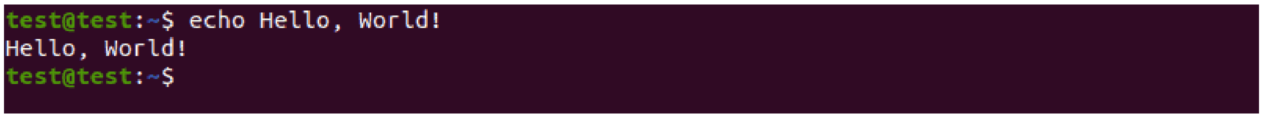
```
 141  man
 142  3a
 143  man intro
 144  man ls
 145  man cat
 146  man man
 147  history
 148  146
 149  history 146
 150  history
 151  clear
 152  history
guru99@VirtualBox:~$ clear
```

```
guru99@VirtualBox:~$
```

8) echo command

The echo command in Linux is used to display a string provided by the user.

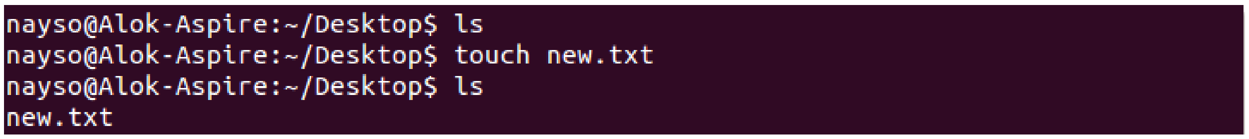
Syntax: echo [string]



```
test@test:~$ echo Hello, World!
Hello, World!
test@test:~$
```

9) touch command

‘touch’ command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, “touch new.txt”.

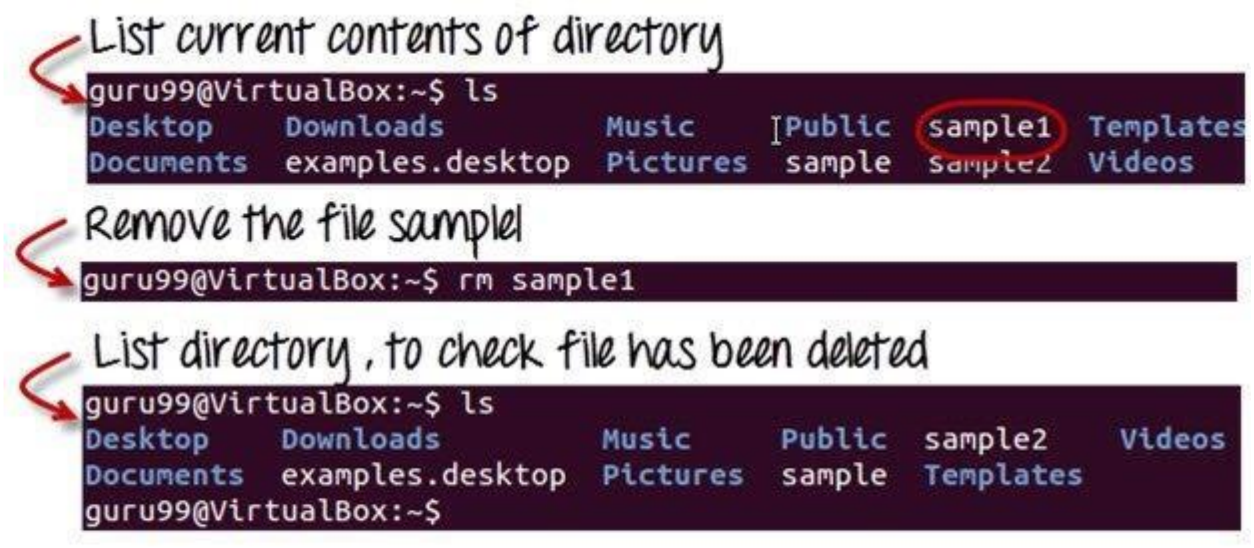


```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ touch new.txt
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
```

10) ‘rm’ command

The ‘rm’ command removes files from the system without confirmation.

Syntax: rm [OPTION]... FILE...


List current contents of directory
Remove the file sample1
List directory , to check file has been deleted

```
guru99@VirtualBox:~$ ls
Desktop     Downloads          Music      Public   sample1  Templates
Documents   examples.desktop   Pictures   sample   sample2  Videos
```

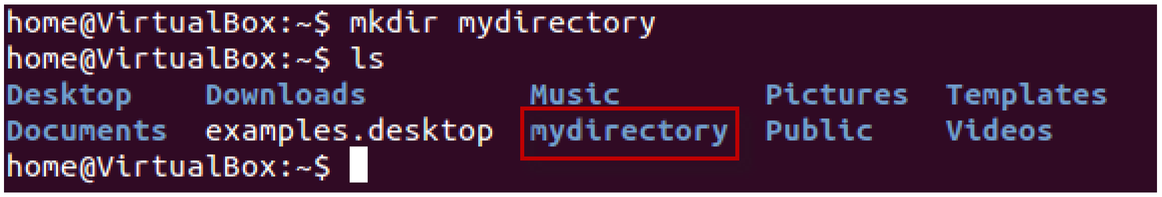
```
guru99@VirtualBox:~$ rm sample1
```

```
guru99@VirtualBox:~$ ls
Desktop     Downloads          Music      Public   sample2    Videos
Documents   examples.desktop   Pictures   sample   Templates
guru99@VirtualBox:~$
```

11) ‘mkdir’ command

Directories can be created using mkdir command.

Syntax: mkdir [options...] [directories ...]

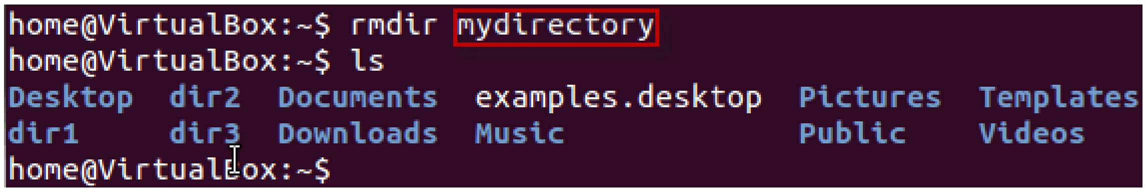


```
home@VirtualBox:~$ mkdir mydirectory
home@VirtualBox:~$ ls
Desktop     Downloads          Music         Pictures   Templates
Documents   examples.desktop   mydirectory   Public     Videos
home@VirtualBox:~$
```

12) ‘rmdir’ command

Directories can be removed using rmdir command.

Syntax: rmdir [-p] [-v | –verbose] [–ignore-fail-on-non-empty] directories …

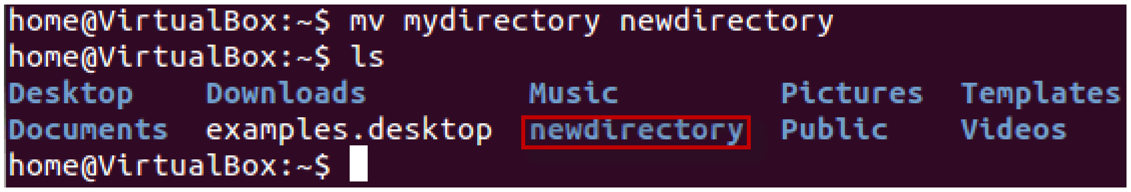


```
home@VirtualBox:~$ rmdir mydirectory
home@VirtualBox:~$ ls
Desktop  dir2  Documents  examples.desktop  Pictures  Templates
dir1     dir3  Downloads  Music             Public    Videos
home@VirtualBox:~$
```

13) ‘mv’ command

The ‘mv’ (move) command can also be used for renaming directories.

Syntax: mv [Option] source destination



```
home@VirtualBox:~$ mv mydirectory newdirectory
home@VirtualBox:~$ ls
Desktop    Downloads         Music         Pictures  Templates
Documents  examples.desktop  newdirectory  Public    Videos
home@VirtualBox:~$
```

14) cd command

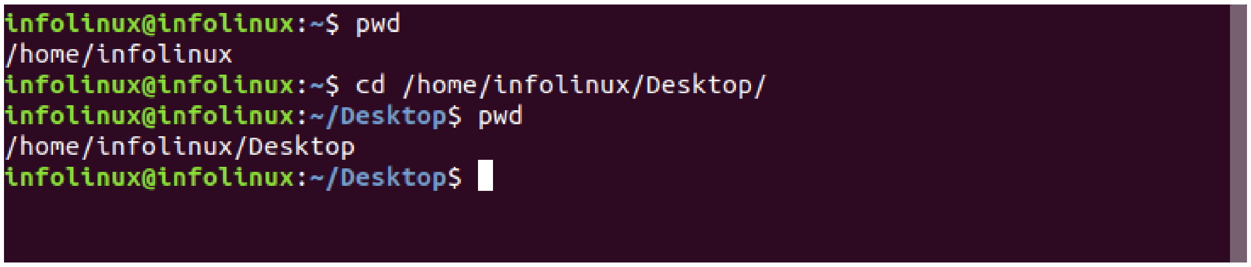
cd stands for Change Directory. It changes the current working directory.

Syntax: $ cd [directory]

Some cd option are shown below:

1. (cd ~) ~ stands for home directory

2. (cd .) . stands for the current directory
3. (cd ..) .. stands for parent directory
4. (cd /) / It takes you to the system's root directory.



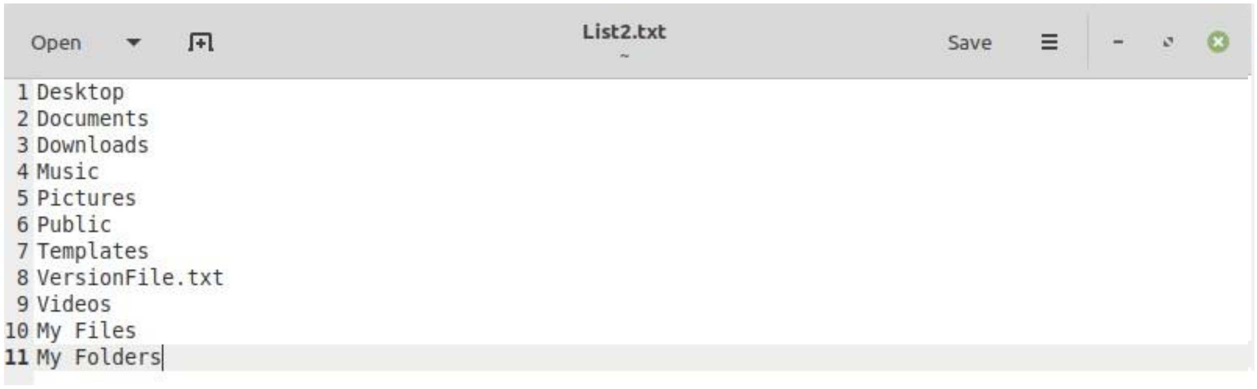
```
infolinux@infolinux:~$ pwd
/home/infolinux
infolinux@infolinux:~$ cd /home/infolinux/Desktop/
infolinux@infolinux:~/Desktop$ pwd
/home/infolinux/Desktop
infolinux@infolinux:~/Desktop$ 
```

15) cmp command

cmp command is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.

For example, we have these text files are shown below: File 1= List.txt

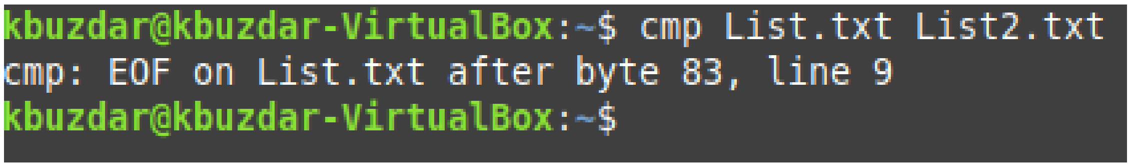
File 2= List2.txt



```
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
VersionFile.txt
Videos
My Files
My Folders
```

Syntax: cmp File1 File2

We have replaced File1 with List.txt and File2 with List2.txt.



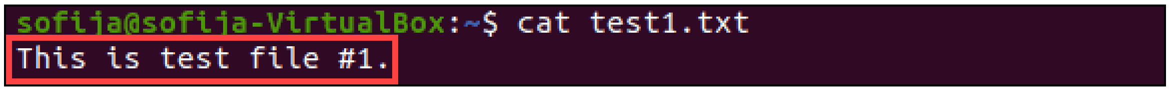
```
kbuzdar@kbuzdar-VirtualBox:~$ cmp List.txt List2.txt
cmp: EOF on List.txt after byte 83, line 9
kbuzdar@kbuzdar-VirtualBox:~$
```

The output of this command reveals that our two specified text files are different from each other.

16) cat command

cat command reads data from the file and gives their content as output. It helps us to create, view, concatenate files.

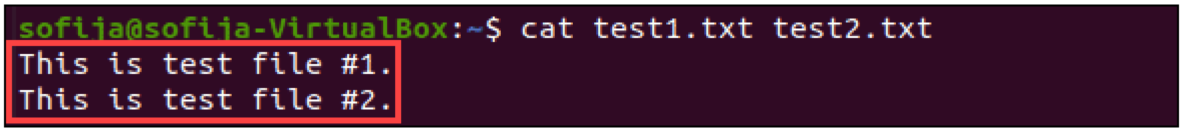
Syntax: Cat test1.txt



```
sofija@sofija-VirtualBox:~$ cat test1.txt
This is test file #1.
```

For multiple files :

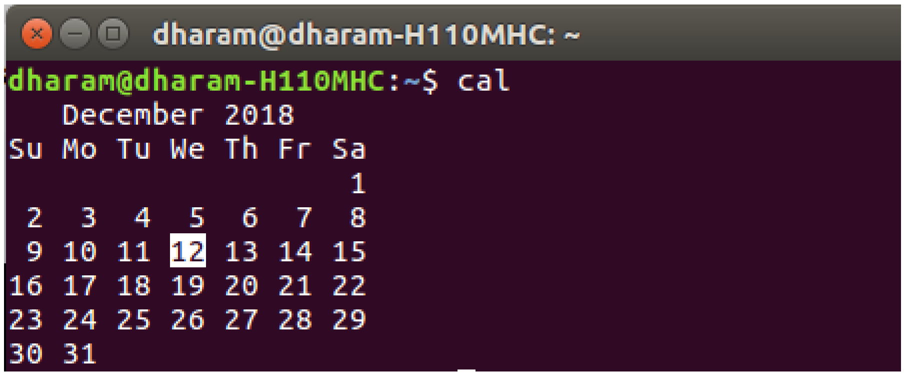
Syntax: Cat test1.txt test2.txt



```
sofija@sofija-VirtualBox:~$ cat test1.txt test2.txt
This is test file #1.
This is test file #2.
```

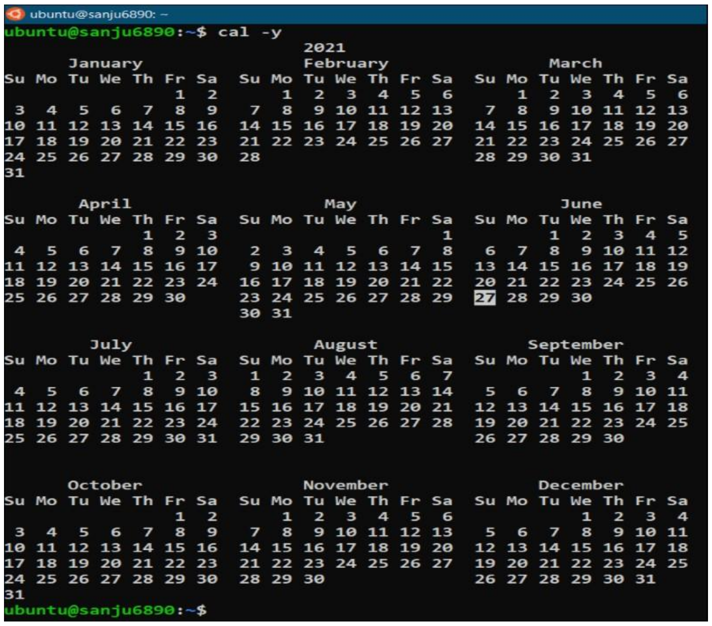
17) cal command

cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.



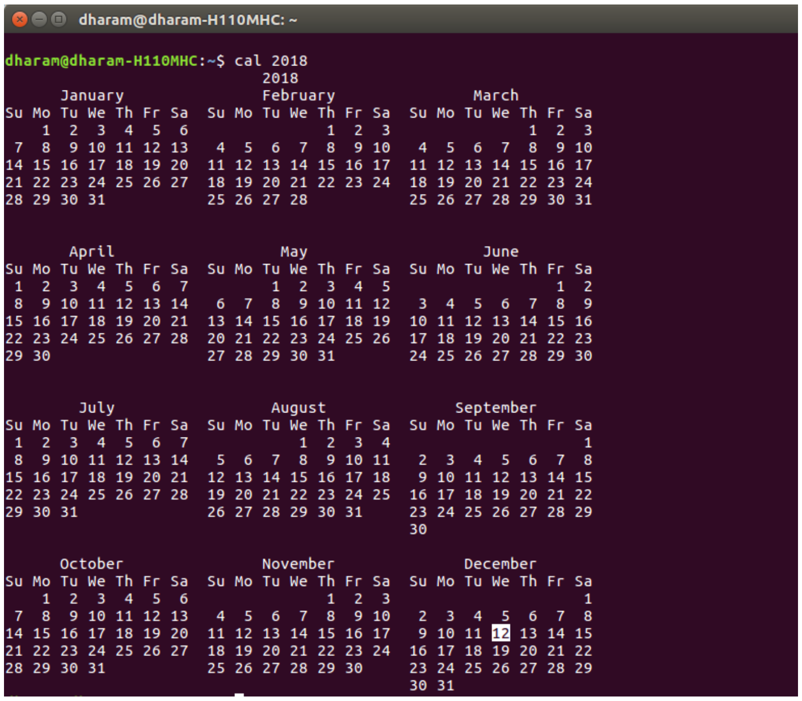
```
dharam@dharam-H110MHC: ~
dharam@dharam-H110MHC:~$ cal
   December 2018
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

cal -y command: Shows the calendar of the complete current year with the current date highlighted.



```
ubuntu@sanju6890: ~
ubuntu@sanju6890:~$ cal -y
                            2021
      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6      1  2  3  4  5  6
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   7  8  9 10 11 12 13
10 11 12 13 14 15 16  14 15 16 17 18 19 20  14 15 16 17 18 19 20
17 18 19 20 21 22 23  21 22 23 24 25 26 27  21 22 23 24 25 26 27
24 25 26 27 28 29 30  28                    28 29 30 31
31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3                     1         1  2  3  4  5
 4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
                      30 31

        July                 August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7            1  2  3  4
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   5  6  7  8  9 10 11
11 12 13 14 15 16 17  15 16 17 18 19 20 21  12 13 14 15 16 17 18
18 19 20 21 22 23 24  22 23 24 25 26 27 28  19 20 21 22 23 24 25
25 26 27 28 29 30 31  29 30 31              26 27 28 29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6            1  2  3  4
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
31
ubuntu@sanju6890:~$
```

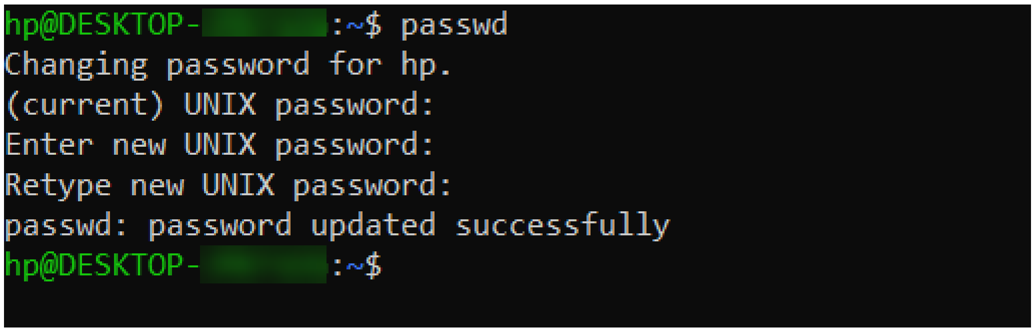
cal 2018 : Shows the whole calendar of the year.



```
dharam@dharam-H110MHC: ~
dharam@dharam-H110MHC:~$ cal 2018
                            2018
      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6               1  2  3               1  2  3
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   4  5  6  7  8  9 10
14 15 16 17 18 19 20  11 12 13 14 15 16 17  11 12 13 14 15 16 17
21 22 23 24 25 26 27  18 19 20 21 22 23 24  18 19 20 21 22 23 24
28 29 30 31           25 26 27 28           25 26 27 28 29 30 31


       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7         1  2  3  4  5                  1  2
 8  9 10 11 12 13 14   6  7  8  9 10 11 12   3  4  5  6  7  8  9
15 16 17 18 19 20 21  13 14 15 16 17 18 19  10 11 12 13 14 15 16
22 23 24 25 26 27 28  20 21 22 23 24 25 26  17 18 19 20 21 22 23
29 30                 27 28 29 30 31        24 25 26 27 28 29 30


        July                 August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4                     1
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   2  3  4  5  6  7  8
15 16 17 18 19 20 21  12 13 14 15 16 17 18   9 10 11 12 13 14 15
22 23 24 25 26 27 28  19 20 21 22 23 24 25  16 17 18 19 20 21 22
29 30 31              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6               1  2  3                     1
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   2  3  4  5  6  7  8
14 15 16 17 18 19 20  11 12 13 14 15 16 17   9 10 11 12 13 14 15
21 22 23 24 25 26 27  18 19 20 21 22 23 24  16 17 18 19 20 21 22
28 29 30 31           25 26 27 28 29 30     23 24 25 26 27 28 29
                                            30 31
```

18) passwd command

passwd command is used to change the user account passwords.



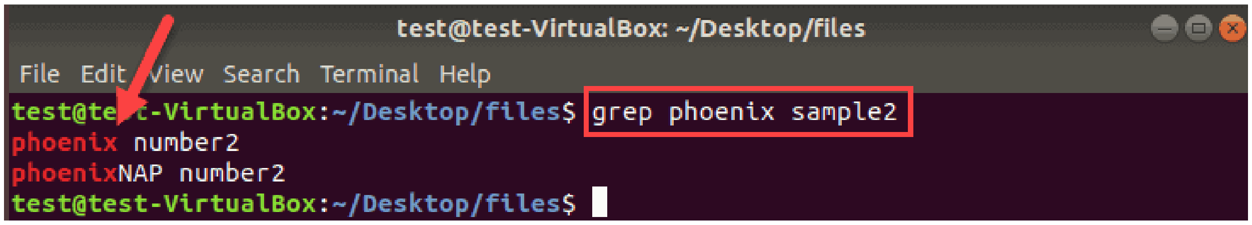
```
hp@DESKTOP-[redacted]:~$ passwd
Changing password for hp.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
hp@DESKTOP-[redacted]:~$
```

19) grep command

grep command used to search for a string of characters in a specified file. The text search pattern is called a regular expression. When it finds a match, it prints the line with the result. The grep command is handy when searching through large log files.

Syntax: grep [options] pattern [files]

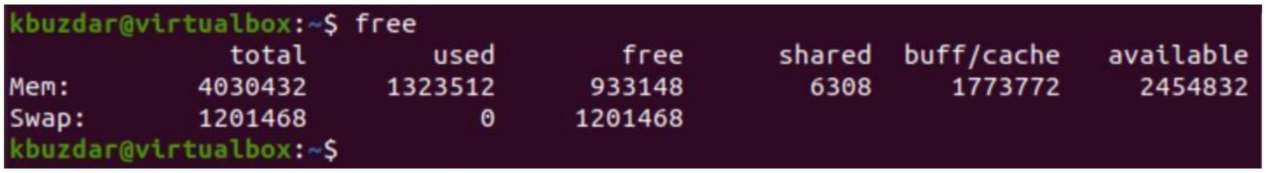
Exa: grep phoenix sample2


test@test-VirtualBox: ~/Desktop/files
File Edit View Search Terminal Help

```
test@test-VirtualBox:~/Desktop/files$ grep phoenix sample2
phoenix number2
phoenixNAP number2
test@test-VirtualBox:~/Desktop/files$
```

20) free command

free command shows the system memory usage (free, used , swaped , cached etc). This field shows the total amount of memory and how much is installed on your system.

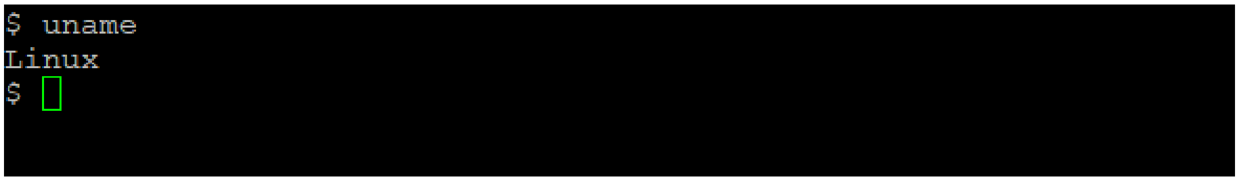


```
kbuzdar@virtualbox:~$ free
              total        used        free      shared  buff/cache   available
Mem:        4030432     1323512      933148        6308     1773772     2454832
Swap:       1201468           0     1201468
kbuzdar@virtualbox:~$
```

21) uname command

The command ‘uname‘ displays the information about the system.

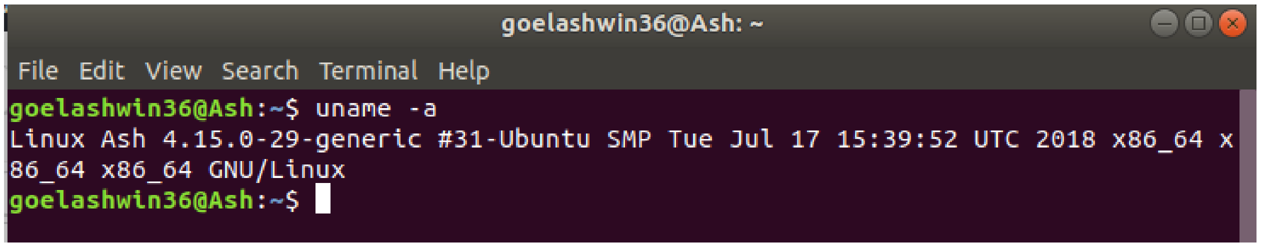
Syntax: **uname [OPTION]**



```
$ uname
Linux
$
```

-a option: It prints all the system information in the following order: Kernel name, network node hostname, kernel release date, kernel version, machine hardware name, hardware platform, operating system.

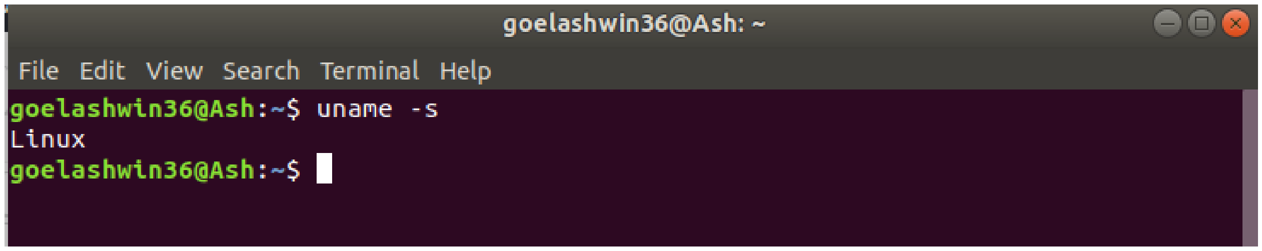
Syntax:$uname   -a



```
goelashwin36@Ash:~$ uname -a
Linux Ash 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018 x86_64 x
86_64 x86_64 GNU/Linux
goelashwin36@Ash:~$
```

-s option: It prints the kernel name.

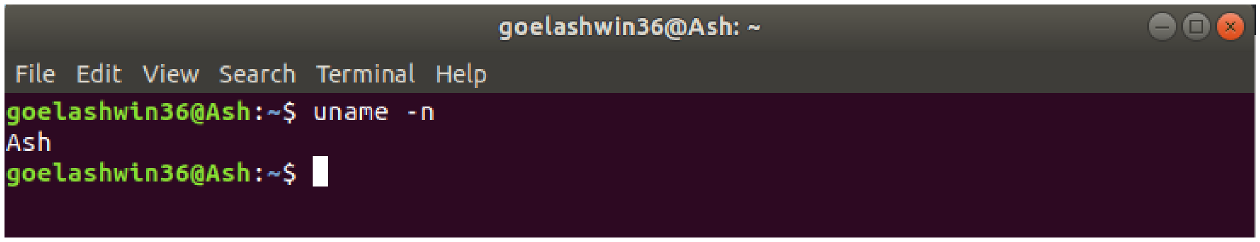
Syntax: $uname   -s



```
goelashwin36@Ash:~$ uname -s
Linux
goelashwin36@Ash:~$
```

-n option: It prints the hostname of the network node (current computer).

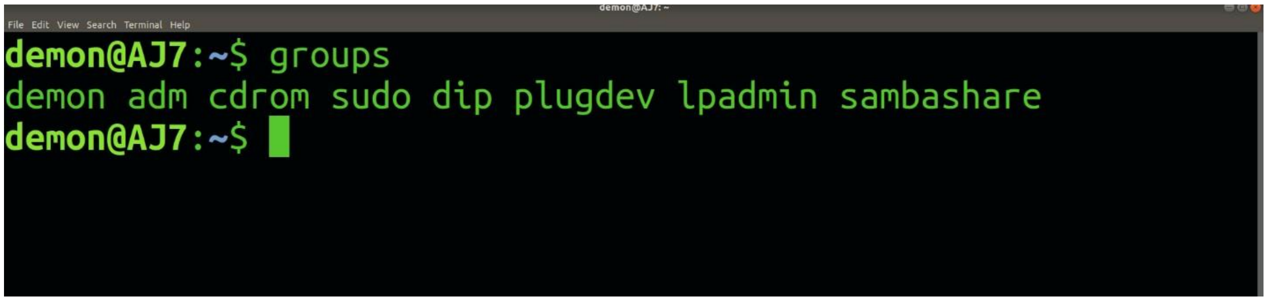
Syntax: $uname   -n



```
goelashwin36@Ash:~$ uname -n
Ash
goelashwin36@Ash:~$
```

22) Group command

Group command displays all the names of group a user is part of.

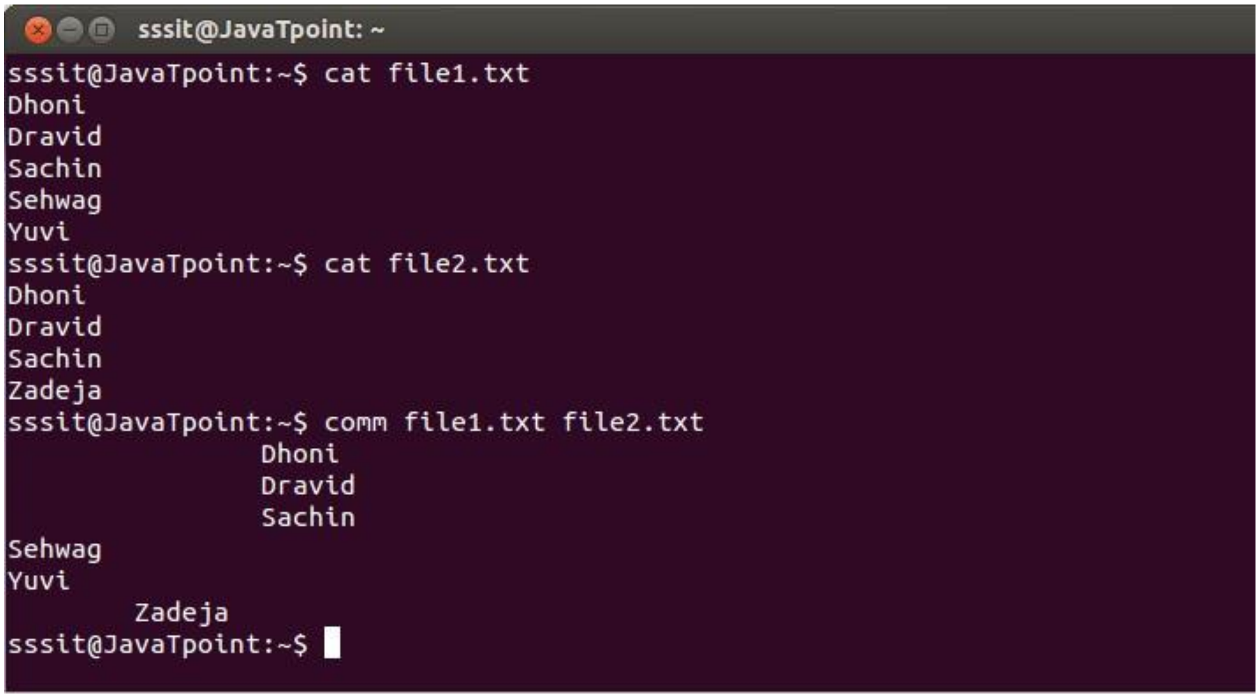


```
demon@AJ7:~$ groups
demon adm cdrom sudo dip plugdev lpadmin sambashare
demon@AJ7:~$ 
```

23) comm commands

The 'comm' command compares two files or streams.

Syntax: comm [file1] [file2]



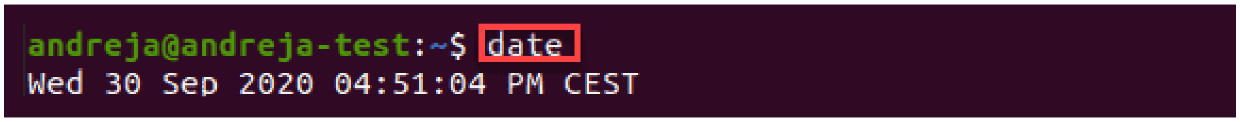
```
sssit@JavaTpoint:~$ cat file1.txt
Dhoni
Dravid
Sachin
Sehwag
Yuvi
sssit@JavaTpoint:~$ cat file2.txt
Dhoni
Dravid
Sachin
Zadeja
sssit@JavaTpoint:~$ comm file1.txt file2.txt
                Dhoni
                Dravid
                Sachin
Sehwag
Yuvi
        Zadeja
sssit@JavaTpoint:~$ 
```

24) date command

date command displays and sets the system date and time. This command also allows users to print the time in different formats and calculate future and past dates.

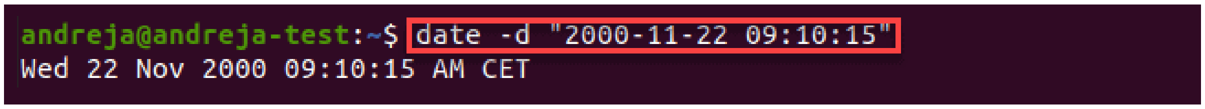
Syntax: `date [option]... [+format]`

To show the current system time and date,



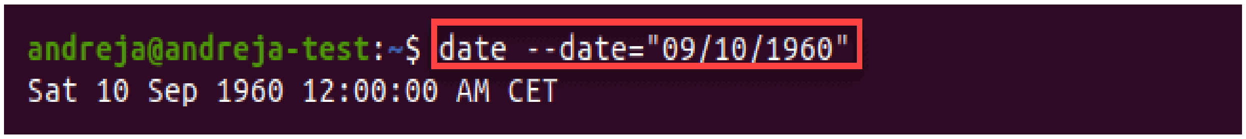
```
andreja@andreja-test:~$ date
Wed 30 Sep 2020 04:51:04 PM CEST
```

-d option: this option allows user to operate on a specific date. For example,



```
andreja@andreja-test:~$ date -d "2000-11-22 09:10:15"
Wed 22 Nov 2000 09:10:15 AM CET
```

--date command: To display the given date string in the format. This command does not affect the system's actual date and time.

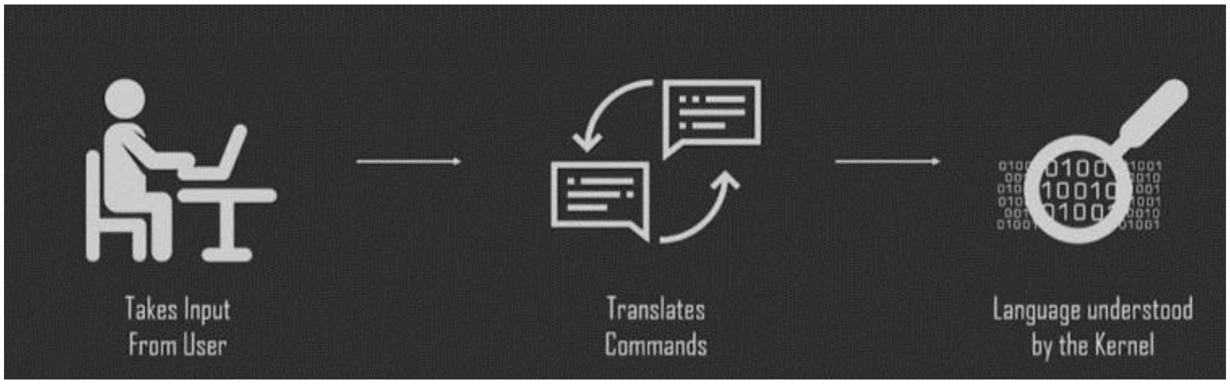


```
andreja@andreja-test:~$ date --date="09/10/1960"
Sat 10 Sep 1960 12:00:00 AM CET
```

II. <u>Shell Scripting</u>

Introduction:

The shell is a command line interpreter. It translates the commands entered by the user and converts them into a language understood by the kernel. Kernel manages resource of Linux O/S. Kernel decides who will use this resource, for how long and when. It runs your programs (or set up to execute binary files).



Computer understand the language of 0's and 1's called binary language, In early days of computing, instruction are provided using binary language, which is difficult for all of us, to read and write. So in O/s there is special program called Shell. Shell accepts your instruction or commands in English and translate it into computers native binary language.

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

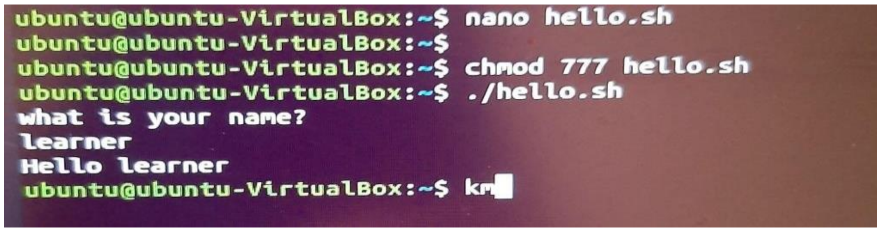
- The Bourne Shell
- The C Shell
- The Korn Shell

- The GNU Bourne-Again Shell

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own

1) **Write a shell script to print your name.**

```
echo "what is your name?"

read PERSON

echo "Hello $PERSON"
```

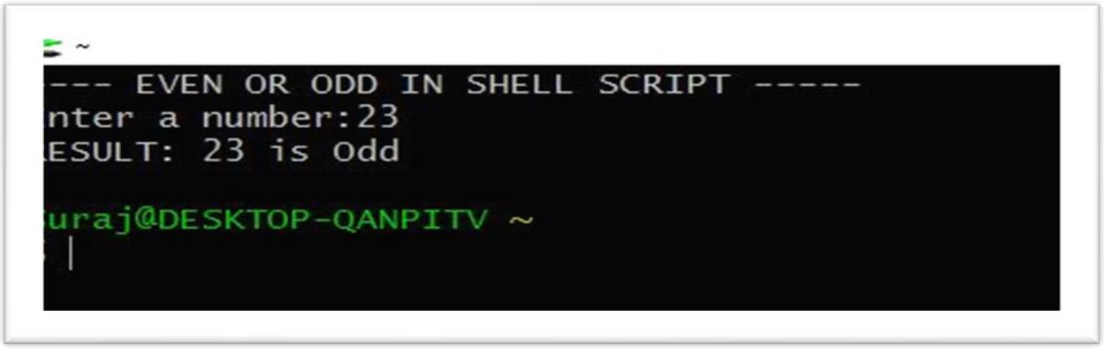


```
ubuntu@ubuntu-VirtualBox:~$ nano hello.sh
ubuntu@ubuntu-VirtualBox:~$
ubuntu@ubuntu-VirtualBox:~$ chmod 777 hello.sh
ubuntu@ubuntu-VirtualBox:~$ ./hello.sh
what is your name?
learner
Hello learner
ubuntu@ubuntu-VirtualBox:~$ km
```

2) Write a shell script to find whether a number is even or odd.

```
clear
echo "---- EVEN OR ODD IN SHELL SCRIPT --------------------"
echo -n "Enter a number: " read n
echo -n "RESULT: "
if [ `expr $n % 2` == 0 ] then
        echo "$n is even"
else
        echo "$n is
fi
```

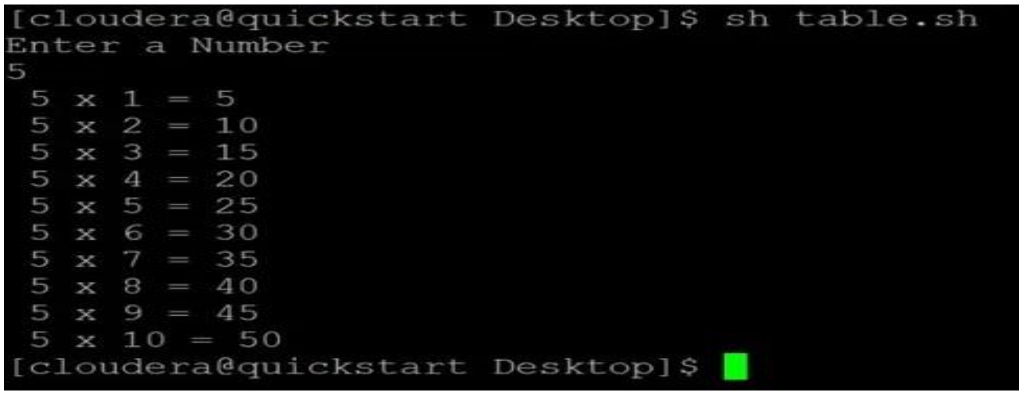
OUTPUT:



```
~
--- EVEN OR ODD IN SHELL SCRIPT -----
nter a number:23
ESULT: 23 is Odd

uraj@DESKTOP-QANPITV ~
 |
```

3) Write a script to print a table of a given number.

```
echo "Enter a Number" read
n
i=1
while [ $i -le 10 ] do
                echo " $n x $i = $(( n * i ))" i=$(( i + 1 ))

done
```

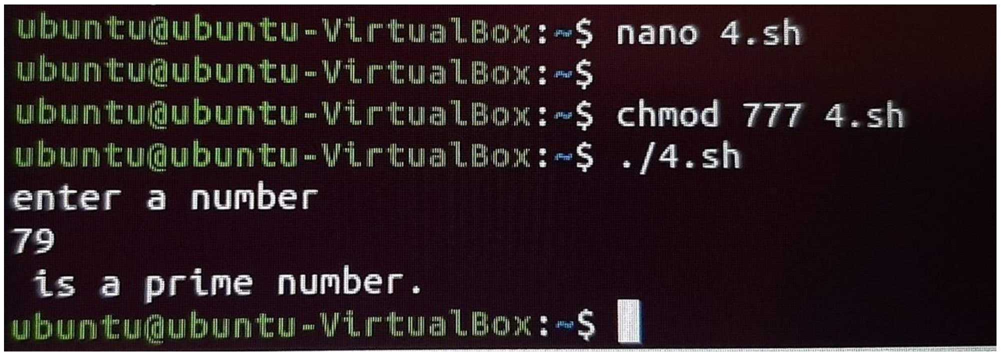


```
[cloudera@quickstart Desktop]$ sh table.sh
Enter a Number
5
 5 x 1 = 5
 5 x 2 = 10
 5 x 3 = 15
 5 x 4 = 20
 5 x 5 = 25
 5 x 6 = 30
 5 x 7 = 35
 5 x 8 = 40
 5 x 9 = 45
 5 x 10 = 50
[cloudera@quickstart Desktop]$ 
```

4) Write a shell script to check whether a given no. is prime or not.

```
echo "enter a number"

read n
for((i=2; i<=num/2; i++)) do
  if [ $((num%i)) -eq 0 ] then
    echo "$num is not a prime number." exit
  fi
done
echo "$num is a prime number."
```

OUTPUT:

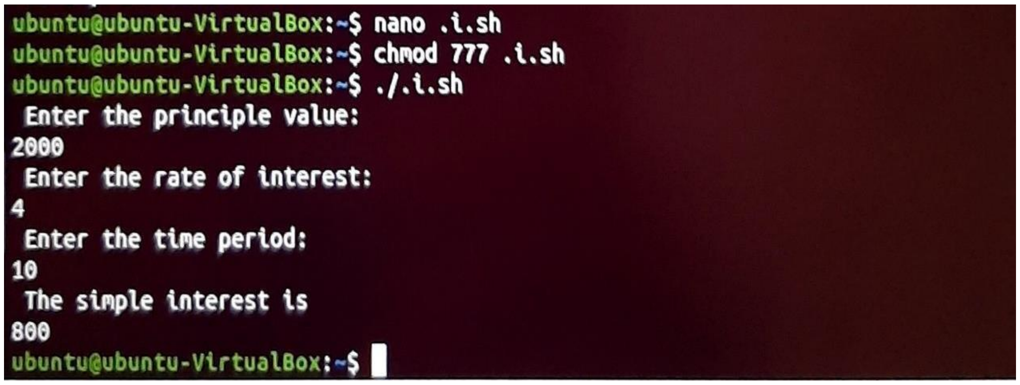


```
ubuntu@ubuntu-VirtualBox:~$ nano 4.sh
ubuntu@ubuntu-VirtualBox:~$
ubuntu@ubuntu-VirtualBox:~$ chmod 777 4.sh
ubuntu@ubuntu-VirtualBox:~$ ./4.sh
enter a number
79
 is a prime number.
ubuntu@ubuntu-VirtualBox:~$
```

5) Write a shell script to find the simple interest.

```
echo " Enter the principle value: " read  p
echo " Enter the rate of interest:" read  r
echo " Enter the time period:" read  t
s=`expr $p \* $t \* $r / 100` echo "
The simple interest is " echo  $s
```

OUTPUT:

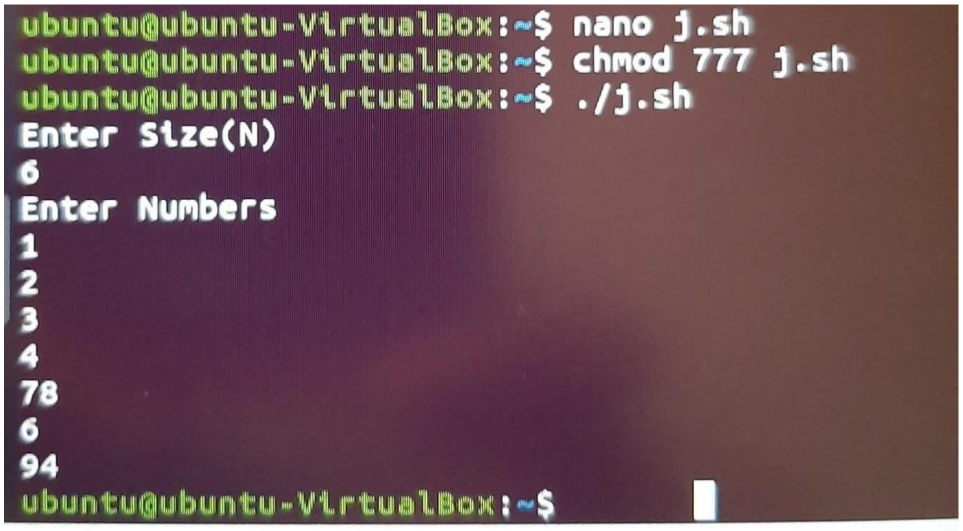


```
ubuntu@ubuntu-VirtualBox:~$ nano .i.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 .i.sh
ubuntu@ubuntu-VirtualBox:~$ ./.i.sh
 Enter the principle value:
2000
 Enter the rate of interest:
4
 Enter the time period:
10
 The simple interest is
800
ubuntu@ubuntu-VirtualBox:~$
```

6) Write a shell script to find sum of n numbers.

```
echo "Enter Size(N)" read  N

sum=0

echo  "Enter Numbers"
for((i=1;i<=N;i++))   do
     read  num                              #get number
     sum=$((sum + num)) #sum+=num
done

echo  $sum
```

OUTPUT:



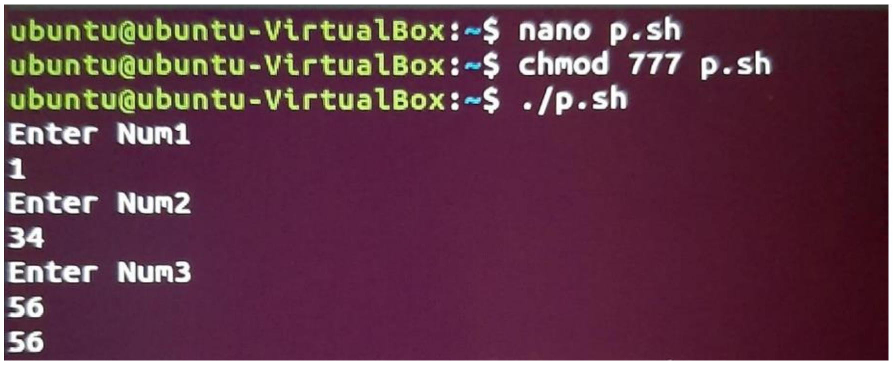
```
ubuntu@ubuntu-VirtualBox:~$ nano j.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 j.sh
ubuntu@ubuntu-VirtualBox:~$ ./j.sh
Enter Size(N)
6
Enter Numbers
1
2
3
4
78
6
94
ubuntu@ubuntu-VirtualBox:~$
```

7. Write a shell script to find the largest number ofthree numbers.

```
echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]then

            echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]then

            echo $num2

else
```

```
            echo $num3
fi
```

OUTPUT:



```
ubuntu@ubuntu-VirtualBox:~$ nano p.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 p.sh
ubuntu@ubuntu-VirtualBox:~$ ./p.sh
Enter Num1
1
Enter Num2
34
Enter Num3
56
56
```

8. **Write a menu driven shell script will point thefollowing menu and execute the give task.**
    a. **Display calender of current month**
    b. **Display today's date and time**
    c. **Display username those are currently logged in thesystem**
    d. **Display your name at given x,y position.**
    e. **Display your terminal number.**

```
echo "Menu"

echo "1. Display calender of current month "echo "2. Display

todays date and time"

echo "3. Display usernames those are currently logged in thesystem"

echo "4. Display your name at given x, y position"echo "5. Display your

terminal number"

echo "6. Exit"

echo "Enter your choice"read c

case $c in

        1)  cal;;
```

2) **date;;**

3) **who;;**

4) **clear**

echo "Enter x, y position"

read x

read y

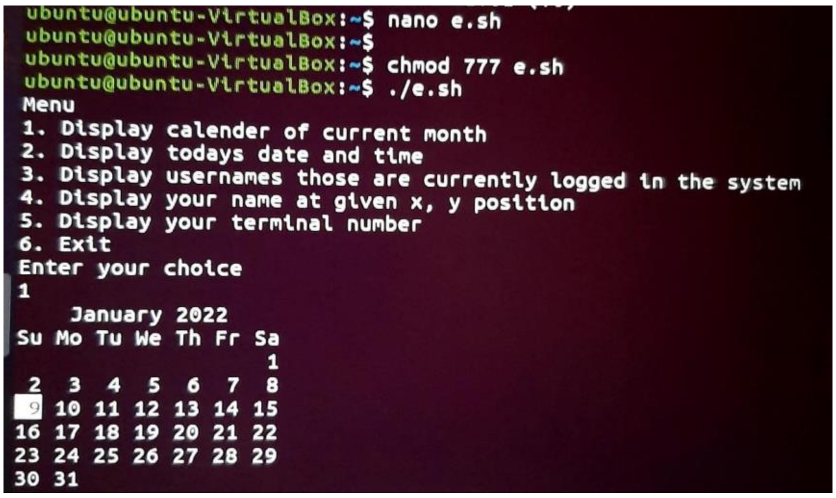
tput cup $x $y

whoami;;

5) **tty;;**

6) **exit**

;;esac



```
ubuntu@ubuntu-VirtualBox:~$ nano e.sh
ubuntu@ubuntu-VirtualBox:~$
ubuntu@ubuntu-VirtualBox:~$ chmod 777 e.sh
ubuntu@ubuntu-VirtualBox:~$ ./e.sh
Menu
1. Display calender of current month
2. Display todays date and time
3. Display usernames those are currently logged in the system
4. Display your name at given x, y position
5. Display your terminal number
6. Exit
Enter your choice
1
    January 2022
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

9. **Write a shell script which will generate first n Fibonacci numbers like :1,1,2,3,5,13,….**

echo "enter

term"

read n

echo "fibonacci series :"

echo -n "$b"

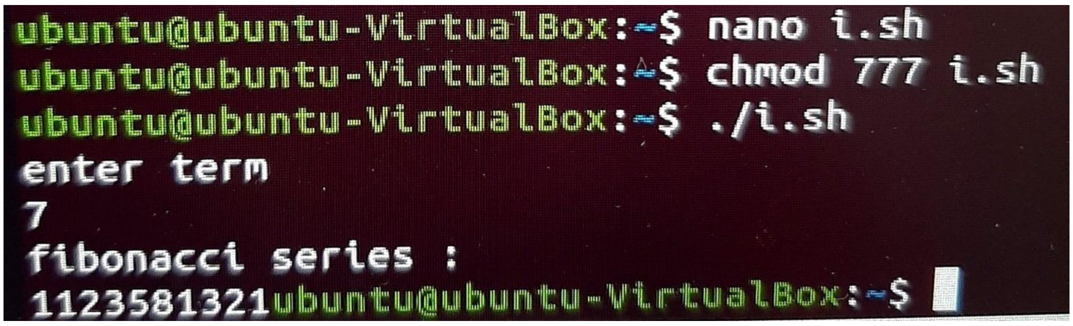
for ((i=1;i<=$n; i++))

do

c=`expr $a +

```
$b`a=$b
b=$c
echo -n "$c"
  done
```

OUTPUT:

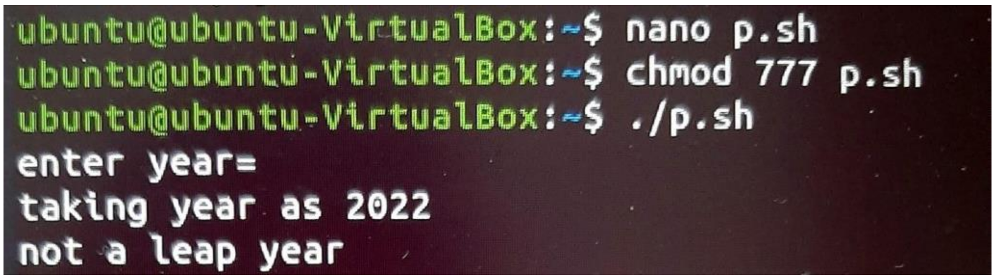


```
ubuntu@ubuntu-VirtualBox:~$ nano i.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 i.sh
ubuntu@ubuntu-VirtualBox:~$ ./i.sh
enter term
7
fibonacci series :
1123581321ubuntu@ubuntu-VirtualBox:~$
```

10. Write a shell script to find whether a given year isleap year or not.

```
leap=$(date +"%Y")
echo taking year as $leap
if [ `expr $leap % 400` -eq 0 ]then

echo leap year
elif [ `expr $leap % 100` -eq 0 ]then

echo not a leap year
elif [ `expr $leap % 4` -eq 0 ]then

echo leap
yearelse

echo not a leap yearfi
```

OUTPUT:

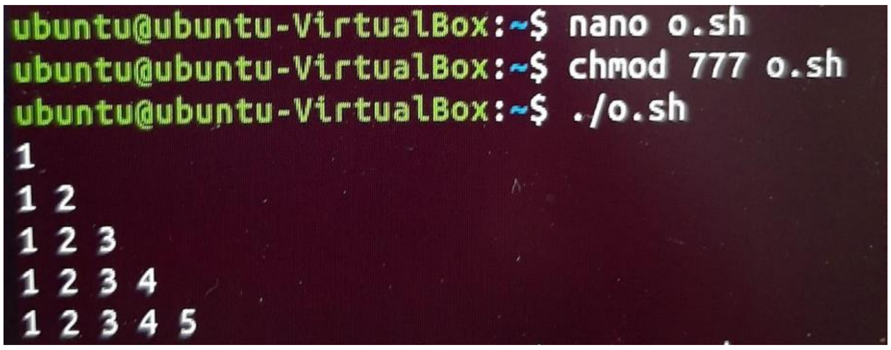


```
ubuntu@ubuntu-VirtualBox:~$ nano p.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 p.sh
ubuntu@ubuntu-VirtualBox:~$ ./p.sh
enter year=
taking year as 2022
not a leap year
```

11. Shell Script to print half pyramid using numbers.

```
number=1
rows=5
for((i=1; i<=rows; i++))
do
        for((j=1; j<=i; j++))
        do
                echo -n "$number "
                number=$((number + 1))
        done
        number=1
done
```

OUTPUT:



```
ubuntu@ubuntu-VirtualBox:~$ nano o.sh
ubuntu@ubuntu-VirtualBox:~$ chmod 777 o.sh
ubuntu@ubuntu-VirtualBox:~$ ./o.sh
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

12. Write a shell script that changes text to upper case

```
#! /bin/bash
echo "Hello World!" |tr 'a-z' 'A-Z'
```

13. Write a shell script to find reverse of given number.

```
#! /bin/bash
echo "Enter a number:"
read n
echo "Entered number is $n"
reversen=0
while [ $n -gt 0 ]
do
r=$(( n % 10 ))
reversen=$(( reversen * 10 + $r ))
n=$(( n / 10 ))
done
echo "Reversed number is $reversen"
```

14. Write a shell script to find sum of floating point numbers.

```
#! /bin/bash
echo "Enter two numbers:"
read num1 num2
echo "The sum of these numbers is: "
echo $num1 + $num2 | bc
```

15. **Write a shell script to make the following operations menu based:**
    a) Addition
    b) Subtraction
    c) Multiplication
    d) Division

```
#! /bin/bash
echo "Enter two numbers:"
read num1 num2
echo $num1 + $num2 | bc
echo $num1 - $num2 | bc
echo $num1 \* $num2 | bc
echo "scale=3;$num1 / $num2" | bc
```

16. Write a shell script to find sum of all digit for given number.

```
#! /bin/bash
echo "Enter a number:"
read n
echo "Entered number is $n"
while [ $n -gt 0 ]
do
x=$(( n % 10 ))
sumn=$(( sumn + $x ))
n=$(( n / 10 ))
done
echo "Sum of digits of number is $sumn"
```

17. Write a shell script to find the factorial of a given no.

```
#! /bin/bash
echo "Enter a number:"
read n
i=1
fact=1
while [ $i -le $n ]
do
fact=$(( fact * i ))
i=$(( i + 1 ))
done
```

18. Write a shell script to find the largest of three numbers and also find the total average.

```
#! /bin/bash
echo "Enter three numbers:"
read n1 n2 n3
largest=$n1
#echo $n1
if [ $n2 -gt $n1 ]
then
largest=$n2
if [ $n3 -gt $n2 ]
then
largest=$n3
fi
fi
total=$(( n1 + n2 +n3 ))
#avg=$(( total / 3 ))
echo "Largest of three is: $largest"
echo "Total of three is: $total"
echo -n "Average of three is: "
echo "scale=2;$total/3" | bc
```

19. Write a shell script which print "invalid no. of arguments" if more than 5 command line arguments otherwise print "valid no. of arguments".

```
#! /bin/bash
echo $1 $2 $3 $4 $5
if [ $# -eq 5 ]
then
echo "Valid arguments"
else
echo "Invalid arguments"
fi
```

20. Write a shell script to find the max. and min. number from the given data set passed by command line argument.

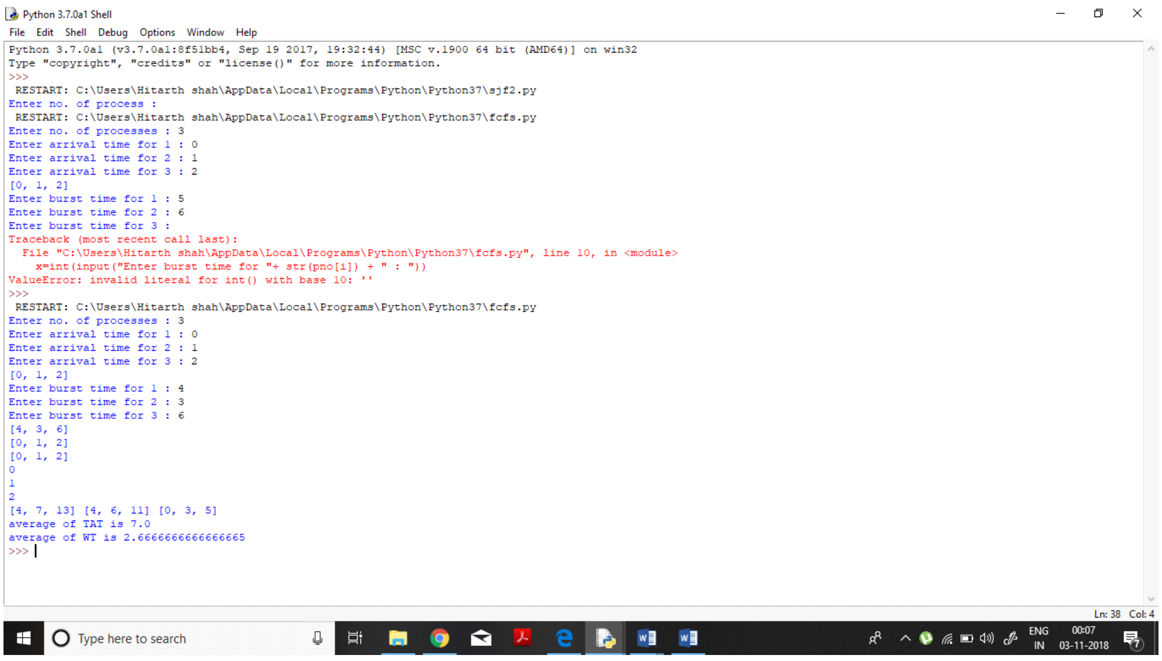
```
#! /bin/bash
#Maximum Value:
echo "Arguments: $*"
max=$1
#echo "Max = $max"
args=("$@")
for(( i=0; i<$#; i++ ))
do
#echo "Values: ${args[i]}"
if [ ${args[i]} -gt $max ]
then
max=${args[i]}
fi
done
echo "Maximum value: $max"
#Minimum Value
min=$1
for(( i=0; i<$#; i++ ))
do
#echo "Values: ${args[i]}"
if [ ${args[i]} -lt $min ]
then
min=${args[i]}
fi
done
echo "Minimum value: $min"
```

III. CPU Scheduling

a) First Come First Serve (FCFS):

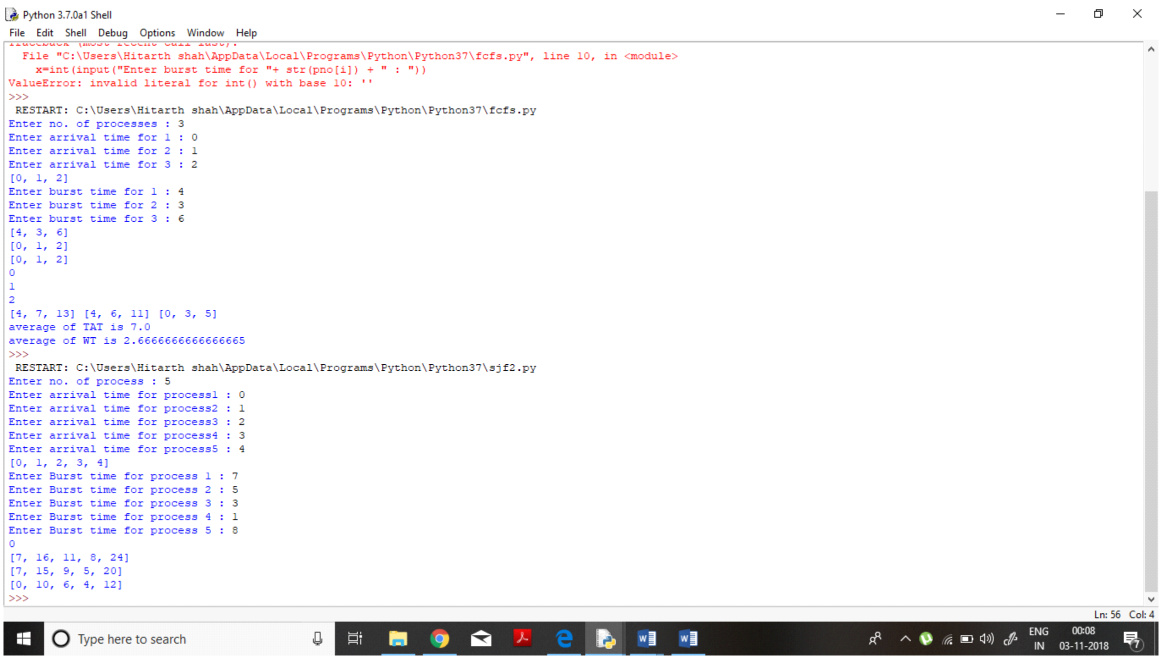
```
pno,at,bt,ct,tat,wt=[], [], [], [], [], []
n=int(input("Enter no. of processes : "))
for i in range(n):
    pno.append(i+1)
for i in range(n):
    x=int(input("Enter arrival time for "+ str(pno[i]) + " : "))
    at.append(x)
print(at)
for i in range(n):
    x=int(input("Enter burst time for "+ str(pno[i]) + " : "))
    bt.append(x)
for i in range(n):
    ct.append(0)
    tat.append(0)
    wt.append(0)
print(bt)
c_bt=0
t_at=at[:]
```

```
print(t_at)
print(at)
max_at=max(at)
f=0
while(f<=(max(pno)-1)):
    t1 = min(t_at)
    print(t1)
    if(t1<=c_bt):
        t2=at.index(t1)
        c_bt+=bt[t2]
        ct[t2]=c_bt
        t_at.remove(t1)
        tat[t2]=ct[t2]-at[t2]
        wt[t2]=tat[t2]-bt[t2]
        if(at[t2]==0):
                at[t2]=1+max_at
        else:
                at[t2]+=max_at
        f+=1
    else:
        c_bt+=1
print(ct, tat, wt)
print("average of TAT is " + str(sum(tat)/float(n)))
print("average of WT is " + str(sum(wt)/float(n)))
```


```
Python 3.7.0a1 (v3.7.0a1:8f51bb4, Sep 19 2017, 19:32:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\sjf2.py
Enter no. of process :
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs.py
Enter no. of processes : 3
Enter arrival time for 1 : 0
Enter arrival time for 2 : 1
Enter arrival time for 3 : 2
[0, 1, 2]
Enter burst time for 1 : 5
Enter burst time for 2 : 6
Enter burst time for 3 :
Traceback (most recent call last):
  File "C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs.py", line 10, in <module>
    x=int(input("Enter burst time for "+ str(pno[i]) + " : "))
ValueError: invalid literal for int() with base 10: ''
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs.py
Enter no. of processes : 3
Enter arrival time for 1 : 0
Enter arrival time for 2 : 1
Enter arrival time for 3 : 2
[0, 1, 2]
Enter burst time for 1 : 4
Enter burst time for 2 : 3
Enter burst time for 3 : 6
[4, 3, 6]
[0, 1, 2]
[0, 1, 2]
0
1
2
[4, 7, 13] [4, 6, 11] [0, 3, 5]
average of TAT is 7.0
average of WT is 2.6666666666666665
>>> 
```

b) Shortest Job First (SJF)

```
pno, at, bt, ct, tat, wt = [], [], [], [], [], []
n=int(input("Enter no. of process : "))
for i in range(n):
    pno.append(i+1)
for i in range(n):
    x=int(input("Enter arrival time for process"+str(pno[i])+" : "))
    at.append(x)
print(at)
for i in range(n):
    x=int(input("Enter Burst time for process "+str(pno[i])+" : "))
    bt.append(x)
    ct.append(0)
    tat.append(0)
    wt.append(0)
c=[]
z=[]
z=bt[:]
c=at[:]
c.sort()
s=c[0]
print(s)
dum=[]
flag=0
tt=0
w=0
while(flag<max(pno)):
    for i in range(n):
            if(at[i]<=s and bt[i]!=0):
                    dum.append(bt[i])
    minbt = min(dum)
    d = bt.index(minbt)
    s = s+minbt
    tt = s - at[d]
    w = tt - bt[d]
    bt[d] = 0
    del ct[d]
    del tat[d]
    del wt[d]
    ct.insert(d,s)
    tat.insert(d,tt)
    wt.insert(d,w)
    flag+=1
    dum=[]
print(ct)
print(tat)
print(wt)
```

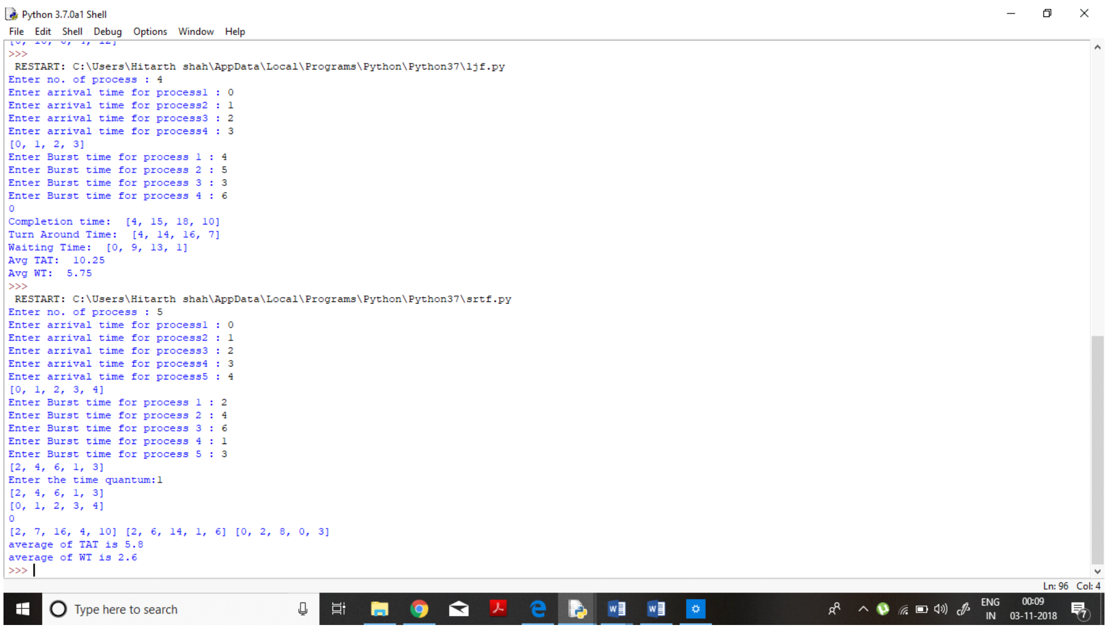


```
 File "C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs.py", line 10, in <module>
   x=int(input("Enter burst time for "+ str(pno[i]) + " : "))
ValueError: invalid literal for int() with base 10: ''
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs.py
Enter no. of processes : 3
Enter arrival time for 1 : 0
Enter arrival time for 2 : 1
Enter arrival time for 3 : 2
[0, 1, 2]
Enter burst time for 1 : 4
Enter burst time for 2 : 3
Enter burst time for 3 : 6
[4, 3, 6]
[0, 1, 2]
[0, 1, 2]
0
1
2
[4, 7, 13] [4, 6, 11] [0, 3, 5]
average of TAT is 7.0
average of WT is 2.6666666666666665
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\sjf2.py
Enter no. of process : 5
Enter arrival time for process1 : 0
Enter arrival time for process2 : 1
Enter arrival time for process3 : 2
Enter arrival time for process4 : 3
Enter arrival time for process5 : 4
[0, 1, 2, 3, 4]
Enter Burst time for process 1 : 7
Enter Burst time for process 2 : 5
Enter Burst time for process 3 : 3
Enter Burst time for process 4 : 1
Enter Burst time for process 5 : 8
0
[7, 16, 11, 8, 24]
[7, 15, 9, 5, 20]
[0, 10, 6, 4, 12]
>>>
```

c) Shortest Remaining Time First (SRTF)

```
pno, at, bt, ct, tat, wt = [], [], [], [], [], []
n=int(input("Enter no. of process : "))
for i in range(n):
    pno.append(i+1)
for i in range(n):
    x=int(input("Enter arrival time for process"+str(pno[i])+" : "))
    at.append(x)
print(at)
for i in range(n):
    x=int(input("Enter Burst time for process "+str(pno[i])+" : "))
    bt.append(x)
    ct.append(0)
    tat.append(0)
    wt.append(0)
print(bt)
tq=int(input("Enter the time quantum:"))
c=[]
z=[]
z=bt[:]
print(z)
c=at[:]
c.sort()
print(c)
s=c[0]
print(s)
ready_queue=[]
flag=0
tt=0
```

```
w=0
j=0
li=[]
while(flag<max(pno)):
    for i in range(n):
        if(at[i]<=s and z[i]!=0):
            ready_queue.append(z[i])
    minbt=min(ready_queue)
    d=z.index(minbt)
    if(minbt>tq):
        s+=tq
        z[d]-=tq
    else:
        s+=tq
        z[d]-=tq
        ct[d]=s
        tat[d]=ct[d]-at[d]
        wt[d]=tat[d]-bt[d]
        flag+=1
    ready_queue=[]
print(ct,tat,wt)
print("average of TAT is " + str(sum(tat)/float(n)))
print("average of WT is " + str(sum(wt)/float(n)))
```

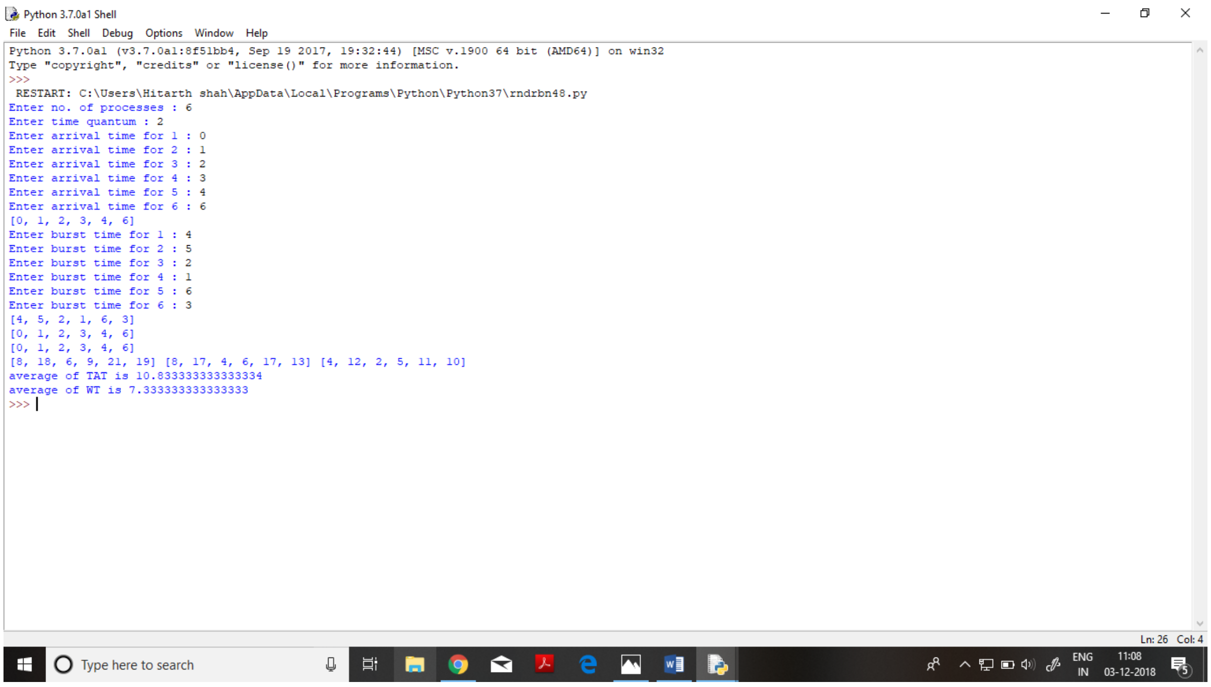


```
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\ljf.py
Enter no. of process : 4
Enter arrival time for process1 : 0
Enter arrival time for process2 : 1
Enter arrival time for process3 : 2
Enter arrival time for process4 : 3
[0, 1, 2, 3]
Enter Burst time for process 1 : 4
Enter Burst time for process 2 : 5
Enter Burst time for process 3 : 3
Enter Burst time for process 4 : 6
0
Completion time:  [4, 15, 18, 10]
Turn Around Time:  [4, 14, 16, 7]
Waiting Time:  [0, 9, 13, 1]
Avg TAT:  10.25
Avg WT:  5.75
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\srtf.py
Enter no. of process : 5
Enter arrival time for process1 : 0
Enter arrival time for process2 : 1
Enter arrival time for process3 : 2
Enter arrival time for process4 : 3
Enter arrival time for process5 : 4
[0, 1, 2, 3, 4]
Enter Burst time for process 1 : 2
Enter Burst time for process 2 : 4
Enter Burst time for process 3 : 6
Enter Burst time for process 4 : 1
Enter Burst time for process 5 : 3
[2, 4, 6, 1, 3]
Enter the time quantum:1
[2, 4, 6, 1, 3]
[0, 1, 2, 3, 4]
0
[2, 7, 16, 4, 10] [2, 6, 14, 1, 6] [0, 2, 8, 0, 3]
average of TAT is 5.8
average of WT is 2.6
>>> 
```

d) Round Robin (RR)

```
pno,at,bt,ct,tat,wt=[],[],[],[],[],[]
n=int(input("Enter no. of processes : "))
tq=int(input("Enter time quantum : "))
```

```
for i in range(n):
      pno.append(i+1)
for i in range(n):
      x=int(input("Enter arrival time for "+ str(pno[i]) + " : "))
      at.append(x)
print(at)
for i in range(n):
      x=int(input("Enter burst time for "+ str(pno[i]) + " : "))
      bt.append(x)
for i in range(n):
      ct.append(0)
      tat.append(0)
      wt.append(0)
print(bt)
'''max_at=max(at)
print(max_at)
min_at=min(at)
print(at.index(min(at)))
print(at)'''
c_bt=0
t_at=at[:]
print(t_at)
print(at)
max_at=max(at)
f=0
ready_queue=[]
t_bt=bt[:]
last=-1
while(f<=(max(pno)-1)):

    for i in range(len(at)):
        if(at[i]<=c_bt and bt[i]!=0 and (i not in ready_queue) and i!=last):
            ready_queue.append(i)
    if(last!=-1):
        ready_queue.append(last)

    if(len(ready_queue)!=0):
        t2=ready_queue[0]
        if(bt[ready_queue[0]]>tq):
            c_bt+=tq
            bt[ready_queue[0]]-=tq
            last=t2
        else:
            c_bt+=bt[ready_queue[0]]
            bt[t2]=0
            ct[t2]=c_bt
            tat[t2]=ct[t2]-at[t2]
            wt[t2]=tat[t2]-t_bt[t2]
            f+=1
            last=-1
        ready_queue=ready_queue[1:]
    else:
        c_bt+=1
print(ct,tat,wt)
print("average of TAT is " + str(sum(tat)/float(n)))
print("average of WT is " + str(sum(wt)/float(n)))
```

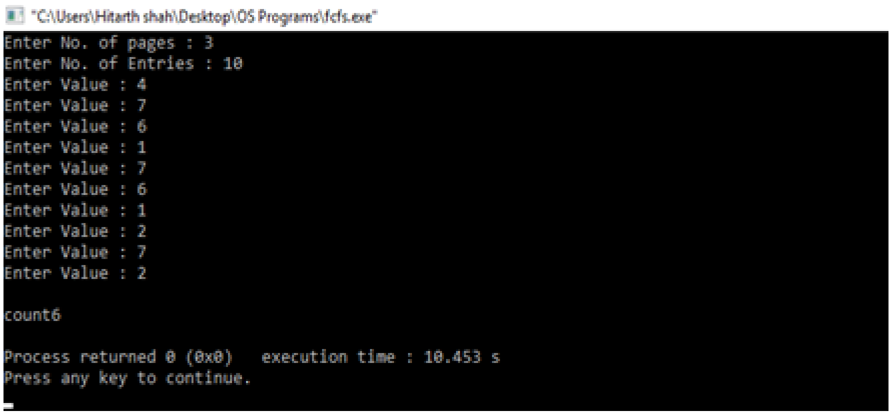


```
Python 3.7.0a1 (v3.7.0a1:8f51bb4, Sep 19 2017, 19:32:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\rndrbn48.py
Enter no. of processes : 6
Enter time quantum : 2
Enter arrival time for 1 : 0
Enter arrival time for 2 : 1
Enter arrival time for 3 : 2
Enter arrival time for 4 : 3
Enter arrival time for 5 : 4
Enter arrival time for 6 : 6
[0, 1, 2, 3, 4, 6]
Enter burst time for 1 : 4
Enter burst time for 2 : 5
Enter burst time for 3 : 2
Enter burst time for 4 : 1
Enter burst time for 5 : 6
Enter burst time for 6 : 3
[4, 5, 2, 1, 6, 3]
[0, 1, 2, 3, 4, 6]
[0, 1, 2, 3, 4, 6]
[8, 18, 6, 9, 21, 19] [8, 17, 4, 6, 17, 13] [4, 12, 2, 5, 11, 10]
average of TAT is 10.833333333333334
average of WT is 7.333333333333333
>>> 
```

IV. Page Replacement

a) First in First Out (FIFO)

```
#include<iostream>
using namespace std;
int main(){
int n,count=0,ent,tmp;
cout<<"Enter No. of pages : ";
cin>>n;
cout<<"Enter No. of Entries : ";
cin>>ent;
int arr[n],arr1[ent];
for(int i=0;i<ent;i++){
    cout<<"Enter Value : ";
    cin>>arr1[i];
}
int j;
for(int k=0;k<n;k++){
    arr[k]=arr1[k];
    cout<<"arr"<<arr[k];
    count++;
}
for(j=n;j<ent;j++){
    tmp=0;
for(int z=0;z<n;z++){
    if(j>n-1 && arr[z]==arr1[j]){
    tmp=1;
    break;
}
}
if(tmp==1){continue;}
else{
    for(int k=1;k<n;k++){
            arr[k-1]=arr[k];
    }
    arr[n-1]=arr1[j];
count++;
}

}
cout<<"\ncount"<<count<<endl;
return 0;
}
```

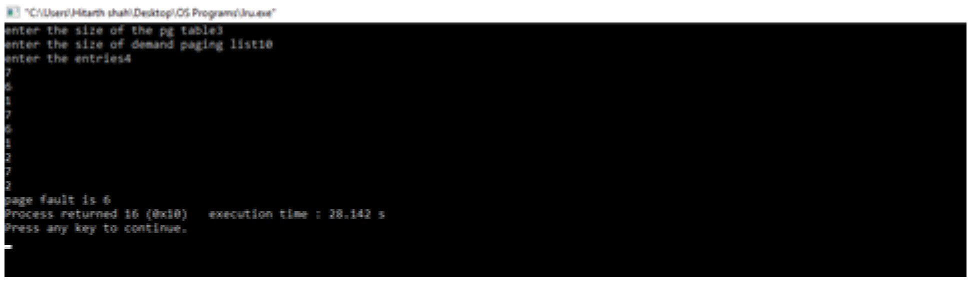


```
"C:\Users\Hitarth shah\Desktop\OS Programs\fcfs.exe"
Enter No. of pages : 3
Enter No. of Entries : 10
Enter Value : 4
Enter Value : 7
Enter Value : 6
Enter Value : 1
Enter Value : 7
Enter Value : 6
Enter Value : 1
Enter Value : 2
Enter Value : 7
Enter Value : 2

count6

Process returned 0 (0x0)   execution time : 10.453 s
Press any key to continue.
```

b) Least Recently Used (LRU)

```
#include<stdio.h>
void main()
{
    int n,i,j,cnt=0,x,val;
    printf("enter the size of the pg table");
    scanf("%d",&n);

    int a[n];
    int f=0,l;
    printf("enter the size of demand paging list");
    scanf("%d",&l);
    int ent[l];
    printf("enter the entries");
    for(i=0;i<l;i++)
    {
            scanf("%d",&ent[i]);
    }
    for(j=0;j<n;j++)
    {
            a[i]=-1;
    }
    for(i=0;i<l;i++)
    {       f=0;
            for(j=0;j<n;j++)
            {
                    if(a[j]==ent[i])
                    {       val=a[j];
                            for(x=j;x<n-1;x++)
                            {
                                    a[x]=a[x+1];
                            }
                            a[x]=val;
                            f=1;
                            break;
                    }
```

```
                }
                if(f==0)
                {
                        for(j=0;j<n-1;j++)
                        {
                                a[j]=a[j+1];
                        }
                        a[j]=ent[i];
                        cnt++;
                }
                else
                        continue;

        }
        printf("page fault is %d ",cnt);
}
```



```
"C:\Users\Hitarth shah\Desktop\OS Programs\lru.exe"
enter the size of the pg table3
enter the size of demand paging list10
enter the entries4
7
6
1
7
6
1
2
7
2
page fault is 6
Process returned 16 (0x10)   execution time : 28.142 s
Press any key to continue.
```

c) Optimal Algorithm

```
#include <iostream>
using namespace std;

// Function to check whether a page exists in a frame or not
bool search(int key, vector<int>& fr)
{
        for (int i = 0; i < fr.size(); i++)
                if (fr[i] == key)
                        return true;
        return false;
}

// Function to find the frame that will not be used
// recently in future after given index in pg[0..pn-1]
int predict(int pg[], vector<int>& fr, int pn, int index)
{
        // Store the index of pages which are going to be used recently in future
        int res = -1, farthest = index;
        for (int i = 0; i < fr.size(); i++) {
                int j;
```

```
            for (j = index; j < pn; j++) {
                if (fr[i] == pg[j]) {
                    if (j > farthest) {
                        farthest = j;
                        res = i;
                    }
                    break;
                }
            }

            // If a page is never referenced in future, return it.
            if (j == pn)
                return i;
        }

        // If all of the frames were not in future, return any of them, we return 0. Otherwise
        // we return res.
        return (res == -1) ? 0: res;
}

void optimalPage(int pg[], int pn, int fn)
{
        // Create an array for given number of frames and initialize it as empty.
        vector<int> fr;
        // Traverse through page reference array and check for miss and hit.
        int hit = 0;
        for (int i = 0; i < pn; i++) {
            // Page found in a frame: HIT
            if (search(pg[i], fr)) {
                hit++;
                continue;
            }
            // Page not found in a frame : MISS
            // If there is space available in frames.
            if (fr.size() < fn)
                fr.push_back(pg[i]);
            // Find the page to be replaced.
            else {
                int j = predict(pg, fr, pn, i + 1);
                fr[j] = pg[i];
            }
        }
        cout << "No. of hits = " << hit << endl;
        cout << "No. of misses = " << pn - hit << endl;
}
```

```
int main()
{
    int pg[] = { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 };
    int pn = sizeof(pg) / sizeof(pg[0]);
    int fn = 4;
    optimalPage(pg, pn, fn);
    return 0;
}
```

Output:

```
No. of hits = 7
No. of misses = 6
```

d) Least Frequently Used

```
#include <iostream.h>
using namespace std;

/* Counts no. of page faults */
int pageFaults(int n, int c, int pages[])
{
    // Initialise count to 0
    int count = 0;
    // To store elements in memory of size c
    vector<int> v;
    // To store frequency of pages
    unordered_map<int, int> mp;

    int i;
    for (i = 0; i <= n - 1; i++) {
            // Find if element is present in memory or not
            auto it = find(v.begin(), v.end(), pages[i]);
            // If element is not present
            if (it == v.end()) {
                    // If memory is full
                    if (v.size() == c) {
                            // Decrease the frequency
                            mp[v[0]]--;
                            // Remove the first element as it is least frequently used
                            v.erase(v.begin());
                    }
                    // Add the element at the end of memory
                    v.push_back(pages[i]);
                    // Increase its frequency
                    mp[pages[i]]++;
```

```
                // Increment the count
                count++;
            }
            else {
                // If element is present, Remove the element and add it at the end
                // Increase its frequency
                mp[pages[i]]++;
                v.erase(it);
                v.push_back(pages[i]);
            }

            // Compare frequency with other pages
            // starting from the 2nd last page
            int k = v.size() - 2;

            // Sort the pages based on their frequency and time at which they arrive
            // if frequency is same then, the page arriving first must be placed first
            while (mp[v[k]] > mp[v[k + 1]] && k > -1) {
                swap(v[k + 1], v[k]);
                k--;
            }
    }
    // Return total page faults
    return count;
}

int main()
{
    int pages[] = { 1, 2, 3, 4, 2, 1, 5 };
    int n = 7, c = 3;
    cout << "Page Faults = " << pageFaults(n, c, pages) << endl;
    cout << "Page Hits = " << n - pageFaults(n, c, pages);
    return 0;
}
```

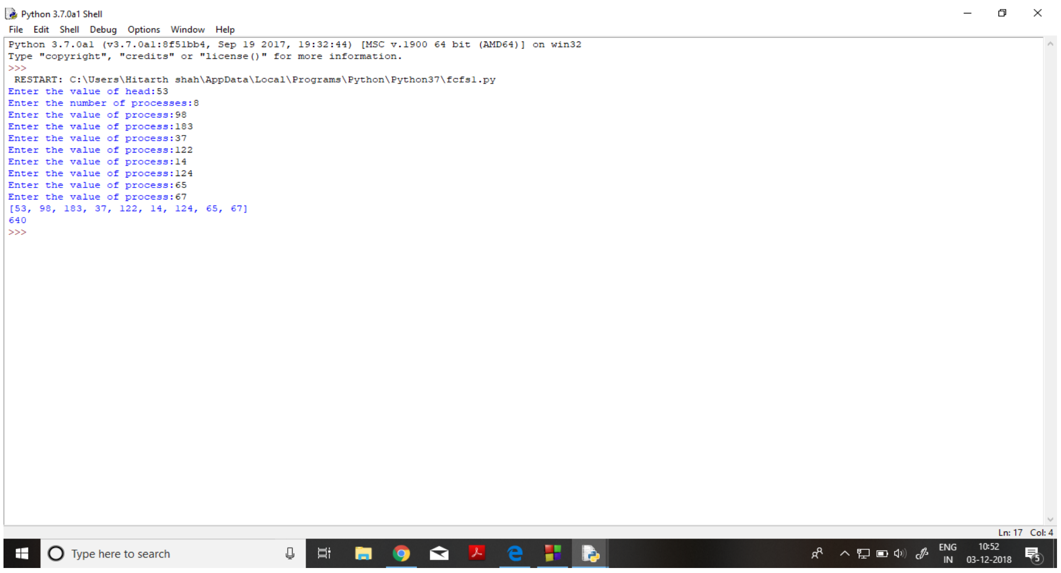
Outputs:

```
Page Faults = 6
Page Hits = 1
```

V. Disk Scheduling

a.) First Come First Serve (FCFS)

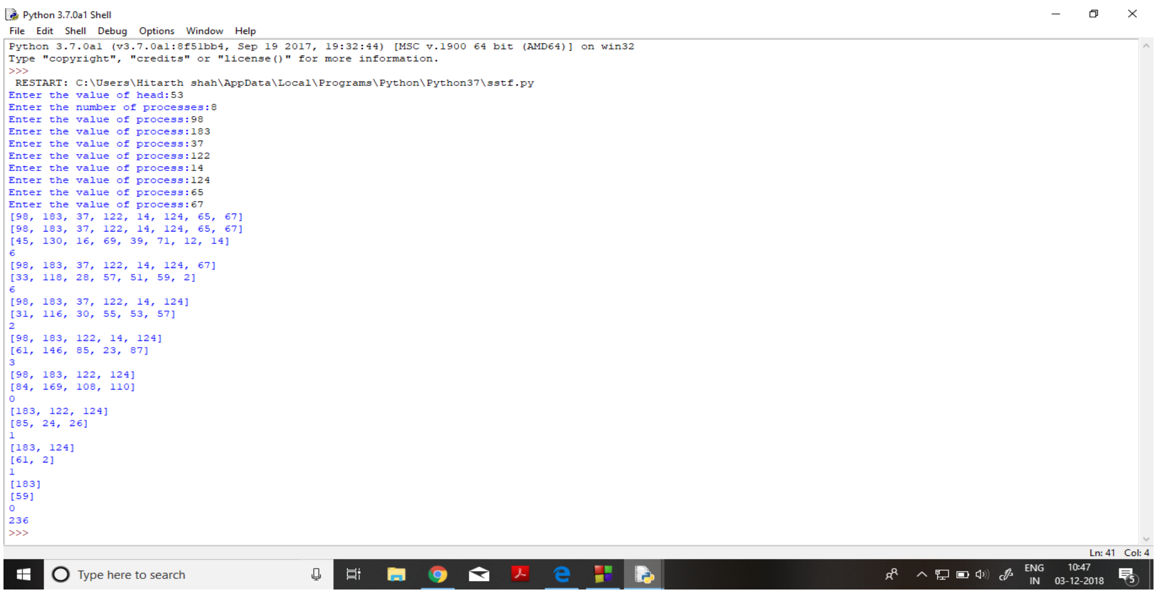
```
head=int(input("Enter the value of head:"))
queue,ans=[],[]
s=0
queue.append(head)
n=int(input("Enter the number of processes:"))
for i in range(n):
    x=int(input("Enter the value of process:"))
    queue.append(x)
print(queue)
for i in range(n):
    if(queue[i+1]>queue[i]):
            j=queue[i+1]-queue[i]
    else:
            j=queue[i]-queue[i+1]
    i+=1
    s=s+j
print(s)
```



```
Python 3.7.0a1 (v3.7.0a1:8f51bb4, Sep 19 2017, 19:32:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\fcfs1.py
Enter the value of head:53
Enter the number of processes:8
Enter the value of process:98
Enter the value of process:183
Enter the value of process:37
Enter the value of process:122
Enter the value of process:14
Enter the value of process:124
Enter the value of process:65
Enter the value of process:67
[53, 98, 183, 37, 122, 14, 124, 65, 67]
640
>>>
```

b.) Shortest Seek Time First (SSTF)

```
head=int(input("Enter the value of head:"))
queue,temp=[],[]
s=0
n=int(input("Enter the number of processes:"))
for i in range(n):
    x=int(input("Enter the value of process:"))
    queue.append(x)
print(queue)
available=queue[:]
for i in range(n):
    temp=[]
    print(available)
    for i in range(len(available)):
        if(head>available[i]):
            ans=head-available[i]
        else:
            ans=available[i]-head
        temp.append(ans)
    print(temp)
    val1=temp.index(min(temp))
    print(val1)
    val=val1+1
    s=s+temp[val1]
    head=available[val1]
    del available[val1]
print(s)
```



```
Python 3.7.0a1 (v3.7.0a1:8f51bb4, Sep 19 2017, 19:32:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Hitarth shah\AppData\Local\Programs\Python\Python37\sstf.py
Enter the value of head:53
Enter the number of processes:8
Enter the value of process:98
Enter the value of process:183
Enter the value of process:37
Enter the value of process:122
Enter the value of process:14
Enter the value of process:124
Enter the value of process:65
Enter the value of process:67
[98, 183, 37, 122, 14, 124, 65, 67]
[98, 183, 37, 122, 14, 124, 65, 67]
[45, 130, 16, 69, 39, 71, 12, 14]
6
[98, 183, 37, 122, 14, 124, 67]
[33, 118, 28, 57, 51, 59, 2]
6
[98, 183, 37, 122, 14, 124]
[31, 116, 30, 55, 53, 57]
2
[98, 183, 122, 14, 124]
[61, 146, 85, 23, 87]
3
[98, 183, 122, 124]
[84, 169, 108, 110]
0
[183, 122, 124]
[85, 24, 26]
1
[183, 124]
[61, 2]
1
[183]
[59]
0
236
>>>
```

c.) C-SCAN Algorithm

```
#include <iostream>
using namespace std;

int size = 8;
int disk_size = 200;

void CSCAN(int arr[], int head)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;

    // appending end values which has to be visited before reversing the direction
    left.push_back(0);
    right.push_back(disk_size - 1);

    // tracks on the left of the head will be serviced when once the head comes back
    // to the beginning (left end).
    for (int i = 0; i < size; i++) {
            if (arr[i] < head)
                    left.push_back(arr[i]);
            if (arr[i] > head)
                    right.push_back(arr[i]);
    }

    // sorting left and right vectors
```

```
std::sort(left.begin(), left.end());
std::sort(right.begin(), right.end());

// first service the requests on the right side of the head.
for (int i = 0; i < right.size(); i++) {
        cur_track = right[i];
        // appending current track to seek sequence
        seek_sequence.push_back(cur_track);

        // calculate absolute distance
        distance = abs(cur_track - head);

        // increase the total count
        seek_count += distance;

        // accessed track is now new head
        head = cur_track;
}

// once reached the right end jump to the beginning.
head = 0;

// adding seek count for head returning from 199 to 0
seek_count += (disk_size - 1);

// Now service the requests again which are left.
for (int i = 0; i < left.size(); i++) {
        cur_track = left[i];

        // appending current track to seek sequence
        seek_sequence.push_back(cur_track);

        // calculate absolute distance
        distance = abs(cur_track - head);

        // increase the total count
        seek_count += distance;

        // accessed track is now the new head
        head = cur_track;
}
cout << "Total number of seek operations = "
        << seek_count << endl;
cout << "Seek Sequence is" << endl;

for (int i = 0; i < seek_sequence.size(); i++) {
```

```
            cout << seek_sequence[i] << endl;
    }
}

int main()
{
    int arr[size] = { 176, 79, 34, 60, 92, 11, 41, 114 };
    int head = 50;
    cout << "Initial position of head: " << head << endl;
    CSCAN(arr, head);
    return 0;
}
```

Output:

```
Initial position of head: 50
Total number of seek operations = 389
Seek Sequence is 60 79 92 114 176 199 0 11 34 41
```

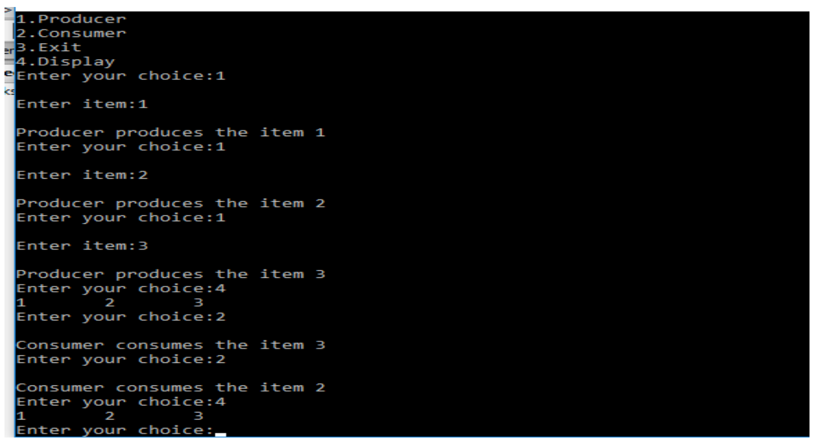
VI. Deadlock and Concurrency

a.) Producer Consumer

```
#include<stdio.h>

int mutex=1,pos=-1,x;
int items[3];
int wait(int s)
{
  return (--s);
}
```

```
int signal(int s)
{
   return(++s);
}

void producer()
{
   mutex=wait(mutex);
   pos=signal(pos);
   if(pos<3)
   {
              printf("\nEnter item:");
      scanf("%d",&x);
     printf("\nProducer produces the item %d",x);
              items[pos]=x;
   }
   mutex=signal(mutex);
}

void consumer()
{
   mutex=wait(mutex);
     pos=wait(pos);
     if(pos>=-1)
   {x=items[pos+1];
     printf("\nConsumer consumes the item %d",x);
   }
   mutex=signal(mutex);
}

void disp()
{
     int i;
     for(i=0;i<=2;i++)
     {        printf(items[i]+" ");}
}

int main()
{
   int n;
   printf("\n1.Producer\n2.Consumer\n3.Exit\n4.Display");
   while(1)
   {
      printf("\nEnter your choice:");
      scanf("%d",&n);
      switch(n)
      {
         case 1:    if((mutex==1)&&(pos<3))
                      producer();
```

```
                else
                   printf("Buffer is full!!");
                break;

          case 2:    if((mutex==1)&&(pos>=0))
                   consumer();
                else
                   printf("Buffer is empty!!");
                break;
                        case 4: disp();
                                           break;
          case 3:
                exit(0);
                break;

       }
    }
    return 0;
}
```



```
1.Producer
2.Consumer
3.Exit
4.Display
Enter your choice:1

Enter item:1

Producer produces the item 1
Enter your choice:1

Enter item:2

Producer produces the item 2
Enter your choice:1

Enter item:3

Producer produces the item 3
Enter your choice:4
1       2       3
Enter your choice:2

Consumer consumes the item 3
Enter your choice:2

Consumer consumes the item 2
Enter your choice:4
1       2       3
Enter your choice:_
```

b.) Banker's algorithm

```
#include<iostream>
using namespace std;

// Number of processes
const int P = 5;

// Number of resources
const int R = 3;

// Function to find the need of each process
void calculateNeed(int need[P][R], int maxm[P][R], int allot[P][R])
{
    // Calculating Need of each P
    for (int i = 0 ; i < P ; i++)
            for (int j = 0 ; j < R ; j++)

                    // Need of instance = maxm instance -allocated instance
                    need[i][j] = maxm[i][j] - allot[i][j];
}

// Function to find the system is in safe state or not
bool isSafe(int processes[], int avail[], int maxm[][R],
                    int allot[][R])
{
    int need[P][R];

    // Function to calculate need matrix
    calculateNeed(need, maxm, allot);

    // Mark all processes as infinish
    bool finish[P] = {0};

    // To store safe sequence
    int safeSeq[P];

    // Make a copy of available resources
    int work[R];
    for (int i = 0; i < R ; i++)
            work[i] = avail[i];

    // While all processes are not finished or system is not in safe state.
    int count = 0;
    while (count < P)
    {
            // Find a process which is not finish and
            // whose needs can be satisfied with current
            // work[] resources.
            bool found = false;
            for (int p = 0; p < P; p++)
            {
                    // First check if a process is finished,
                    // if no, go for next condition
                    if (finish[p] == 0)
                    {
                            // Check if for all resources of
                            // current P need is less
                            // than work
                            int j;
                    for (j = 0; j < R; j++)
                        if (need[p][j] > work[j])
                            break;

                    // If all needs of p were satisfied.
                    if (j == R)
                    {
                        // Add the allocated resources of
                        // current P to the available/work
                        // resources i.e.free the resources
                        for (int k = 0 ; k < R ; k++)
                            work[k] += allot[p][k];

                        // Add this process to safe sequence.
                        safeSeq[count++] = p;

                        // Mark this p as finished
                        finish[p] = 1;

                        found = true;
                    }
                }
            }

            // If we could not find a next process in safe sequence.
            if (found == false)
            {
                cout << "System is not in safe state";
                return false;
            }
    }

    // If system is in safe state then safe sequence will be as below
    cout << "System is in safe state.\nSafe"
            " sequence is: ";
    for (int i = 0; i < P ; i++)
            cout << safeSeq[i] << " ";

    return true;
}

int main()
{
    int processes[] = {0, 1, 2, 3, 4};

    // Available instances of resources
    int avail[] = {3, 3, 2};

    // Maximum R that can be allocated to processes
    int maxm[][R] = {{7, 5, 3},
                     {3, 2, 2},
                     {9, 0, 2},
                     {2, 2, 2},
                     {4, 3, 3}};

    // Resources allocated to processes
    int allot[][R] = {{0, 1, 0},
                      {2, 0, 0},
                      {3, 0, 2},
                      {2, 1, 1},
                      {0, 0, 2}};

    // Check system is in safe state or not
    IsSafe(processes, avail, maxm, allot);
    return 0;
}
```

Output:

```
System is in safe state.
Safe sequence is: 1 3 4 0 2
```

Date: 14th March, 2022 **Prepared By:**

Dr. Sonam Nahar and Dr. Rutvij H. Jhaveri
(CSE Department)