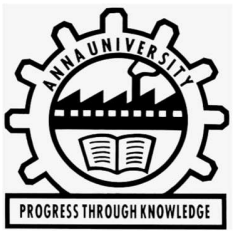
DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044



BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by

Mr/Ms. _____ of Register Number _____

for the _____ laboratory during the period

March 2022 to June 2022.

Date: **Lab In-charge**

INDEX

EXP NO: 1 DATE : 11-03-2022	BASIC UNIX COMMANDS

AIM;

To implement the basic Unix/Linux Commands in Ubuntu Terminal.

I. Basic commands:

1. $date: displays the current date and time
2. $cal MM YYYY: displays the mentioned month and year's calendar
3. $whoami: displays who's logged in
4. $who: displays the user online, time logged in

```
dcl46@dcl46-Veriton-Series:~$ date
Friday 11 March 2022 02:04:37 AM IST
dcl46@dcl46-Veriton-Series:~$ cal 08 2022
    August 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

dcl46@dcl46-Veriton-Series:~$ whoami
dcl46
dcl46@dcl46-Veriton-Series:~$ who
dcl46    :0           2022-03-11 18:51 (:0)
```

5. $uptime: displays the current uptime of the system
6. $w: displays the information about the logged in user

```
dcl46@dcl46-Veriton-Series:~$ uptime
 02:14:31 up 56 min,  1 user,  load average: 0.72, 0.96, 0.95
dcl46@dcl46-Veriton-Series:~$ w
 02:18:49 up  1:00,  1 user,  load average: 0.77, 0.80, 0.88
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
dcl46    :0       :0               18:51    ?xdm?  19:53   0.00s /usr/lib/gdm3/g
```

7. $uname -a : shows kernel information:

```
dcl46@dcl46-Veriton-Series:~$ uname -a
Linux dcl46-Veriton-Series 5.13.0-30-generic #33~20.04.1-Ubuntu SMP Mon Feb 7 14:25:10 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

8. $man command name: shows how the command mentioned works:

9. $df- shows the disk usage

```
dcl46@dcl46-Veriton-Series:~$ man who
dcl46@dcl46-Veriton-Series:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             1874256        0   1874256   0% /dev
tmpfs             381548     1724    379824   1% /run
/dev/sda6      169780592 14198164 146888312   9% /
tmpfs            1907724    35380   1872344   2% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            1907724        0   1907724   0% /sys/fs/cgroup
/dev/loop1         63488    63488         0 100% /snap/core20/1361
/dev/loop0         56960    56960         0 100% /snap/core18/2284
/dev/loop3           128      128         0 100% /snap/bare/5
/dev/loop6        224256   224256         0 100% /snap/gnome-3-34-1804/72
/dev/loop2         63488    63488         0 100% /snap/core20/1328
/dev/loop5        168832   168832         0 100% /snap/gnome-3-28-1804/161
```

10. $du – shows the directory space usage

```
dcl46@dcl46-Veriton-Series:~$ du
44      ./Documents/1
156     ./Documents
4       ./.mozilla/extensions
4       ./.mozilla/firefox/Pending Pings
4       ./.mozilla/firefox/Crash Reports/events
44      ./.mozilla/firefox/Crash Reports
4       ./.mozilla/firefox/8vojs56v.default-release/extensions
1360    ./.mozilla/firefox/8vojs56v.default-release/gmp-gmpopenh264/1.8.1.1
1364    ./.mozilla/firefox/8vojs56v.default-release/gmp-gmpopenh264
324     ./.mozilla/firefox/8vojs56v.default-release/sessionstore-backups
4       ./.mozilla/firefox/8vojs56v.default-release/features
8       ./.mozilla/firefox/8vojs56v.default-release/bookmarkbackups
```

11.$passwd- changed the current password:

```
dcl46@dcl46-Veriton-Series:~$ passwd
Changing password for dcl46.
Current password:
New password:
```

II. Directory management:

1. $mkdir dir_name: creates a new directory with the name mentioned
2. $cd dir_name: goes to the directory mentioned
3. $cd .. : returns back to the parent directory
4. $cd : returns to home
5. $pwd: shows the path of current working directory

```
dcl46@dcl46-Veriton-Series:~$ mkdir newfold
dcl46@dcl46-Veriton-Series:~$ cd newfold
dcl46@dcl46-Veriton-Series:~/newfold$ mkdir newfold1
dcl46@dcl46-Veriton-Series:~/newfold$ cd newfold1
dcl46@dcl46-Veriton-Series:~/newfold/newfold1$ cd..
cd..: command not found
dcl46@dcl46-Veriton-Series:~/newfold/newfold1$ cd ..
dcl46@dcl46-Veriton-Series:~/newfold$ pwd
/home/dcl46/newfold
dcl46@dcl46-Veriton-Series:~/newfold$ cd
dcl46@dcl46-Veriton-Series:~$
```

6. $rmdir dir_name: removes the directory mentioned

```
dcl46@dcl46-Veriton-Series:~/newfold$ rmdir newfold1
dcl46@dcl46-Veriton-Series:~/newfold$ cd newfold1
bash: cd: newfold1: No such file or directory
dcl46@dcl46-Veriton-Series:~/newfold$
```

7. $rm * : removes all the files in a directory
8. $mv olddir newdir : move an old directory into a new directory

```
dcl46@dcl46-Veriton-Series:~/newfold$ ls
1.c
dcl46@dcl46-Veriton-Series:~/newfold$ rm *
dcl46@dcl46-Veriton-Series:~/newfold$ ls
dcl46@dcl46-Veriton-Series:~/newfold$ cd
dcl46@dcl46-Veriton-Series:~$ mkdir newfold1
dcl46@dcl46-Veriton-Series:~$ mv newfold newfold1
dcl46@dcl46-Veriton-Series:~$ cd newfold1
dcl46@dcl46-Veriton-Series:~/newfold1$ ls
newfold
dcl46@dcl46-Veriton-Series:~/newfold1$
```

III. File management commands:

1. $ls: displays all the file names in the current directory
2. $ls -l : displays all files and its details like size, etc.
3. $ls -a : displays all files including hidden ones
4. $touch filename: creates a new file with the name mentioned
5. $gedit filename: opens text editor to edit the file mentioned
6. $more filename: displays the content of the file
7. $cp orgfilename newfilename: copies content of orginal file to the new file
8. $mv filename1 filename2 : renames the orginal file name to the new file name
9. $rm filename: removes the mentioned file

```
dcl46@dcl46-Veriton-Series:~/newfold$ cd
dcl46@dcl46-Veriton-Series:~$ mkdir newfold1
dcl46@dcl46-Veriton-Series:~$ mv newfold newfold1
dcl46@dcl46-Veriton-Series:~$ cd newfold1
dcl46@dcl46-Veriton-Series:~/newfold1$ ls
newfold
dcl46@dcl46-Veriton-Series:~/newfold1$ cd newfold
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ touch file1.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ touch file2.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ touch file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ ls
file1.txt  file2.txt  file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ ls -l
total 0
-rw-rw-r-- 1 dcl46 dcl46 0 Mar 11 02:48 file1.txt
-rw-rw-r-- 1 dcl46 dcl46 0 Mar 11 02:48 file2.txt
-rw-rw-r-- 1 dcl46 dcl46 0 Mar 11 02:48 file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ ls -a
.  ..  file1.txt  file2.txt  file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ gedit file1.txt
^C
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file1.txt
CS6108
OPERATING SYSTEMS

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ cp file1.txt file2.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file2.txt
CS6108
OPERATING SYSTEMS

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ mv file1.txt file4.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file4.txt
CS6108
OPERATING SYSTEMS

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ rm file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ ls
file2.txt  file4.txt
```

10. $wc file_name : count of the total number of lines, words, and characters contained in a file.
11. $cat filename1 filename2: concatenates the content of file2 into file1 and prints it all

```
 3  3 27 file2.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file2.txt
CS6108
OPERATING SYSTEMS

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ wc file2.txt
 3  3 27 file2.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ cat file2.txt file4.txt
CS6108
OPERATING SYSTEMS

CS6108
OPERATING SYSTEMS

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$
```

12. $cat >filename : write into the file
13. $echo random_text : prints the random text entered

```
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ touch file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ cat >file3.txt
welcome
^C
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file3.txt
welcome
```

14. $head filename: prints the first few lines of a file
15. $tail filename- prints the last few lines of a file
16. $head – n filename : Displays the first n lines of the file
17. $tail –n filename: Displays the last n lines of a file

```
d Thunderbird Mail lton-Series:~$ head p2.c
#include <stdio.h>
#include<string.h>
struct student
{
int rollno;
char name[60];
}s1;
void main()
{
s1.rollno=1;
dcl46@dcl46-Veriton-Series:~$ tail p2.c
int rollno;
char name[60];
}s1;
void main()
{
s1.rollno=1;
strcpy(s1.name, “intellipaat”);
printf( "Rollno : %d\n", s1.rollno);
printf( "Name : %s\n", s1.name);
}
```

```
dcl46@dcl46-Veriton-Series:~$ head -2 p2.c
#include <stdio.h>
#include<string.h>
dcl46@dcl46-Veriton-Series:~$ tail -3 p2.c
printf( "Rollno : %d\n", s1.rollno);
printf( "Name : %s\n", s1.name);
}
```

IV. Extract, Sort, Search and Filter Data commands:

1. $grep pattern filename : searches for the given pattern in the file mentioned
i) $grep pattern filename -v : prints all the lines that do not match the given pattern
ii)$grep pattern filename -n: prints matched line and line number
iii) $grep pattern filename -l: prints only the names of file with matching lines
iv) $grep pattern filename -c: prints the count of matching lines

```
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file2.txt
CS6108
OPERATING SYSTEMS
CS6106
Database management systems

Computer architecture

Graph theory

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ grep systems file2.txt
Database management systems
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ grep systems file2.txt -v
CS6108
OPERATING SYSTEMS
CS6106

Computer architecture

Graph theory

dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ grep systems file2.txt -n
4:Database management systems
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ grep systems file2.txt -l
file2.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ grep systems file2.txt -c
1
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$
```

2. $sort filename – arranges the file alphabetically or numerically
i) $sort filename -n : sorts numerically irrespective of blanks, new lines
ii) $sort filename -r : reverses sorting order

```
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file1.txt
EVS
DBMS
LA
GT
CA
OS
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ sort file1.txt
CA
DBMS
EVS
GT
LA
OS
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ sort file1.txt -r
OS
LA
GT
EVS
DBMS
CA
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ sort file1.txt -n
CA
DBMS
EVS
GT
LA
OS
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$
```

V. File permission commands:

1. chmod +rwx filename : add permissions(read,write,executing if programs)
2. chmod +x filename : allows executable permissions
3. chmod -wx filename : removes write and executing permissions

```
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ more file3.txt
welcome
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ chmod -wx file3.txt
dcl46@dcl46-Veriton-Series:~/newfold1/newfold$ cat >file3.txt
bash: file3.txt: Permission denied
```

Couldn't execute gedit filename because writing permission has been removed

VI. Shell commands:

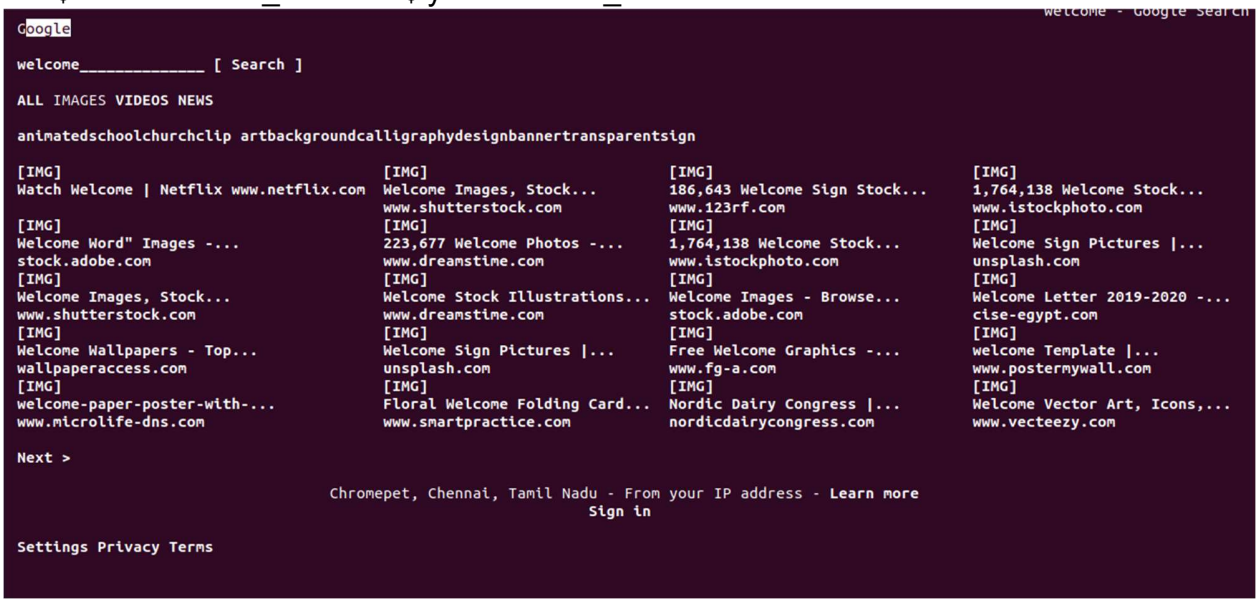
1. $echo $SHELL – displays which shell will be executed automatically when logged in
2. $ cat /etc/shells – shows all the available shells
3. $which bash – displays the path for the executable bash/shell executed current*ly*

```
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ echo $SHELL
/bin/bash
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ which bash
/usr/bin/bash
```

4.$sudo apt install pkgname : installs the mentioned packages
packages installed here include: links2,lynx

```
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ sudo apt install links2
Reading package lists... Done
Building dependency tree
Reading state information... Done
links2 is already the newest version (2.20.2-1).
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
```

-> $links2 domain_name or $lynx2 domain_name



5.$sudo apt-cache search pkgname/anything related installpkg:searches and displays the package to be installed

```
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ sudo apt-cache search CentOS
[sudo] password for dcl46:
apt-dater - terminal-based remote package update manager
apt-dater-host - host helper application for apt-dater
drbl - diskless remote boot, and a disk cloning utility
libappimage-dev - Development files for libappimage
libappimage0 - Core library for appimage
oz - install virtual machine guest OSs with minimal input the user
rinse - RPM installation environment
vuls - Vulnerability scanner for Linux/FreeBSD, agentless, written in Go
xen-tools - Tools to manage Xen virtual servers
```

6. $ctrl +F1/F2/F3 till F7 : displays the various consoles
7. $chmod 777 filename1 : gives all read,write and execute permission
8. $chmod -777 filename1 : removes all read,write and execute permission

```
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ chmod 777 p1.c
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ cat >file1.txt
hii^C
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ chmod -777 p1.c
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$ cat >p1.c
bash: p1.c: Permission denied
dcl46@dcl46-Veriton-Series:~/Desktop/2020503540/temp$
```

RESULT:

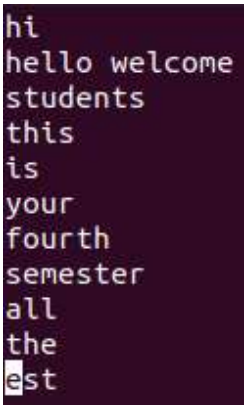
Hence various UNIX commands have been executed successfully

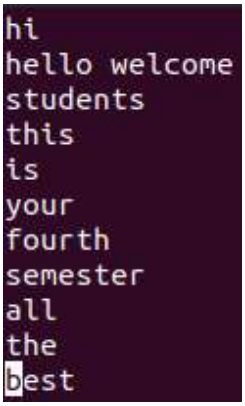
EXP NO: 2 DATE : 18-03-2022	FAMILIARIZATION WITH VIM COMMANDS

AIM:

To implement the vim commands using vim editor.

1) u - to undo the last the command and U to undo the whole line

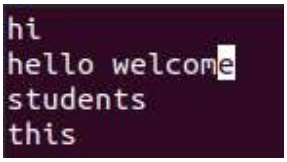




2) CTRL-R to redo

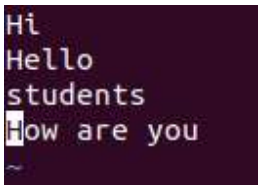
Before:

After:

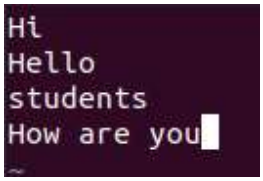


3) A - to append text at the end

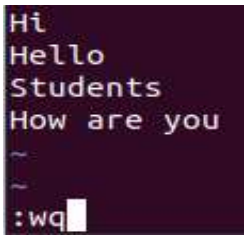
Before:



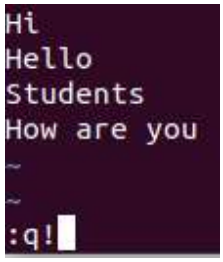
After:



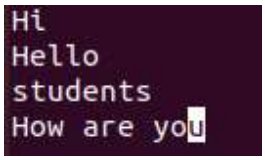
4) :wq - to save and exit



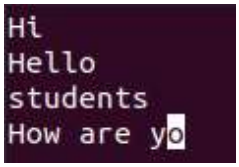
5) :q! - to trash all changes



6) dw - move the cursor to the beginning of the word to delete that word Before:



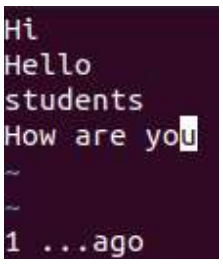
After:



7) 2w - to move the cursor two words forward.

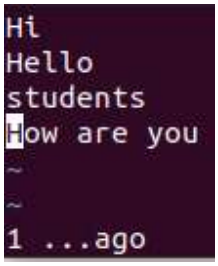
Before:

After:



8) 3e - to move the cursor to the end of the third word forward.

Before:



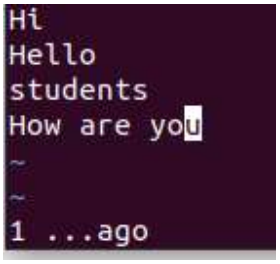
After:

9) x - to delete the unwanted character

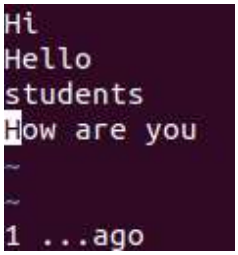
10)x - to delete the unwanted character

11)0 (zero) to move to the start of the line.

Before:

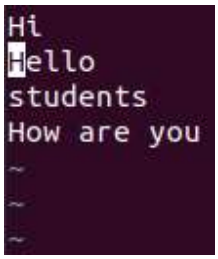


After:

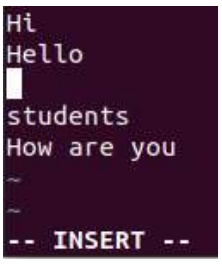


12)o - opens a line below the cursor and starts Insert mode.

Before:



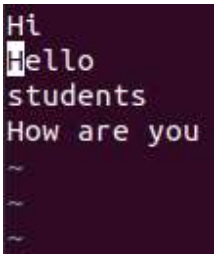
After:

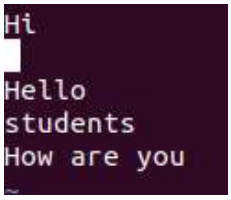


13)O - opens a line above the cursor

Before:

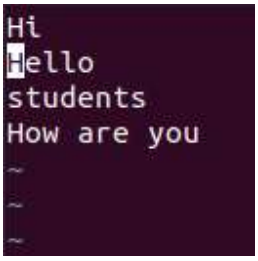
After:



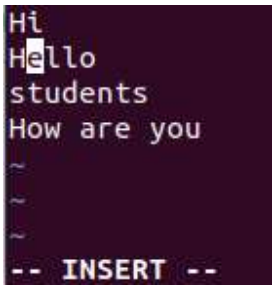


14)a - inserts text after the cursor.

Before:

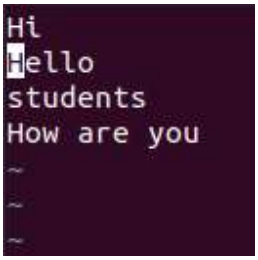


After:

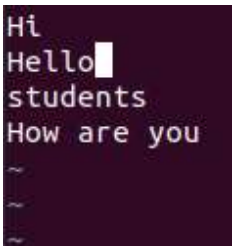


15)A - inserts text after the end of the line.

Before:

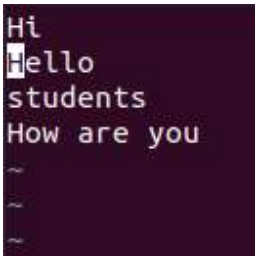


After:

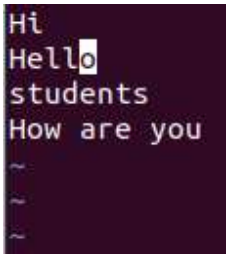


16)e - command moves to the end of a word.

Before:

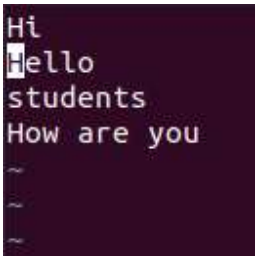


After:

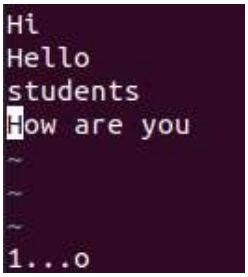


17)G - to move you to the bottom of the file.

Before:



After:



VIM PROGRAMS :

1) Writing and executing Hello World program in vim editor

```
#include<iostream>
using namespace std;
int main()
{
        cout<<"\nHello World \n";
        return 0;
}
```

OUTPUT:

```
multicore@multicore-Veriton-M4660G:~/Documents$ g++ prog1.cpp
multicore@multicore-Veriton-M4660G:~/Documents$ ./a.out

Hello World
```

2) Executing Basic arithmetic operations (addition, subtraction, multiplication, division, remainder) in vim editor.

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

int main() {
    int a, b;
    cout<<"\nEnter number 1: ";
    cin>>a;
    cout<<"\nEnter number 1: ";
    cin>>b;
    cout << "a + b = " << (a + b) << endl;
    cout << "a - b = " << (a - b) << endl;
    cout << "a * b = " << (a * b) << endl;
    cout << "a / b = " << (a / b) << endl;
    cout << "a % b = " << (a % b) << endl;
    return 0;
}
```

OUTPUT:

```
multicore@multicore-Veriton-M4660G:~/Documents$ g++ prog2.cpp
multicore@multicore-Veriton-M4660G:~/Documents$ ./a.out

Enter number 1: 10

Enter number 1: 5
a + b = 15
a - b = 5
a * b = 50
a / b = 2
a % b = 0
```

RESULT :

Hence vim commands have been successfully executed.

EXP NO: 3 DATE : 01-04-2022	SHELL SCRIPTING

1. Write a shell script to convert seconds in HH:MM:SS (24 hours format). Read the input from the CLI argument and perform operations only using loops and control structures.

AIM:

To write a shell script to convert seconds in HH:MM:SS (24 hours format).

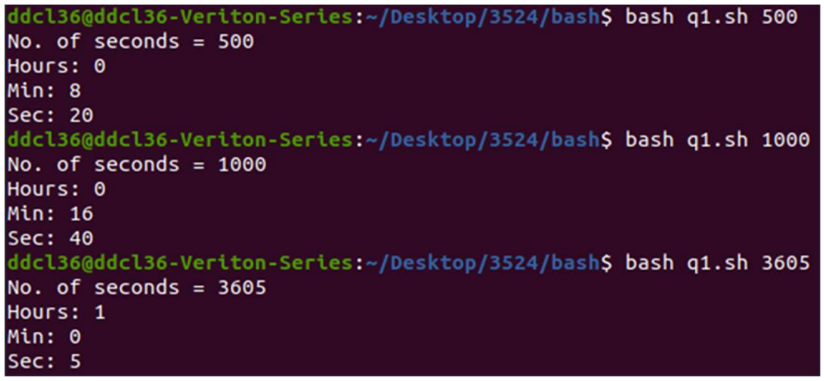
ALGORITHM:

Step 1: Start
Step 2: Read the number of seconds as input from the user.
Step 3: Find the number of hours elapsed by repeatedly subtracting using a loop.
Step 4: Find the number of minutes elapsed by repeatedly subtracting using a loop.
Step 5: Take the remaining seconds as the seconds elapsed.
Step 6: Stop

PROGRAM CODE:

```
#!/bin/bash
echo "No. of seconds = $1"
((tmp = $1))
((hrs = 0))
((min = 0))
((sec = 0))
while [ $tmp -ge 3600 ]
do
        ((hrs = hrs+1))
        ((tmp = tmp-3600))
done
while [ $tmp -ge 60 ]
do
        ((min = min+1))
        ((tmp = tmp-60))
done
((sec = tmp))
((hrs = hrs%24))
echo "Hours: $hrs"
echo "Min: $min"
echo "Sec: $sec"
~
```

OUTPUT:



RESULT:
Hence, a shell script was written to convert a given number of seconds into the respective hours, minutes and seconds.

2. Write a shell program for temperature conversion (i.e) [Fahrenheit to Celsius and Celsius to Fahrenheit]. Read the input from the prompting user.

AIM: To write a shell program for temperature conversion between Fahrenheit and Celsius

ALGORITHM:
Step 1: Start
Step 2: Ask the user on the type of conversion to be performed.
Step 3: Read the value of temperature to be converted.
Step 4: Apply the appropriate formula for the conversion to take place.
Step 5: Print the result.
Step 6: Stop

PROGRAM CODE:

```
#!/bin/bash
echo "1. Farenheit to Celsius"
echo "2. Celsius to Farenheit"
echo "Enter your choice:"
read g
res=0
echo "Enter the value to be converted"
read x
if [ $g -eq 1 ];
then
        ((res= ($x-32)*5/9))
elif [ $g -eq 2 ];
then
        ((res= (9*$x)/5 + 32))
fi
echo "Result: $res"
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$ bash q2.sh
1. Farenheit to Celsius
2. Celsius to Farenheit
Enter your choice:
2
Enter the value to be converted
35
Result: 95
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$ bash q2.sh
1. Farenheit to Celsius
2. Celsius to Farenheit
Enter your choice:
1
Enter the value to be converted
95
Result: 35
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$
```

RESULT:

Hence, a shell script has been written to convert a given temperature from Celsius to Fahrenheit and vice versa.

3. Write a shell program to print the number of vowels, consonants, digits in a given string and read input from the CLI argument.

AIM: To write a shell program to print the number of vowels, consonants, digits in a given string.

ALGORITHM:

Step 1: Start
Step 2: Read the string from the user.
Step 3: Make use of the $grep command to match the characters with appropriate patterns.
Step 4: Make use of the $wc command to find the count of such occurrences.
Step 5: Print the respective counts.
Step 6: Stop

PROGRAM CODE:

```
#!/bin/bash
echo "String: $1"

(( digCount = $(echo $1 | grep -o "[0-9]" | wc --lines) ))
(( vowCount = $(echo $1 | grep -o "[aeiouAEIOU]" | wc --lines) ))
(( conCount = $(echo $1 | grep -o "[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]"
 | wc --lines) ))

echo "There are $vowCount vowels, $conCount consonants, and $digCount digits."
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$ bash q3.sh ABCdef01
String: ABCdef01
There are 2 vowels, 4 consonants, and 2 digits.
```

RESULT:

Hence, a shell script has been written to find the count of vowels, consonants and digits in any given string.

4. Write a program to find the sum of digits in the given integer using shell script. Read the input on prompting the user.

AIM: To write a shell script to find the sum of digits in a given integer.

ALGORITHM:

Step 1: Start
Step 2: Read the integer from the user and store it in a variable x.
Step 3: Initialize a variable s and assign the value 0 to it.
Step 4: Pass the variable x through a loop such that it runs as long as x is greater tha 0.
Step 5: Add the remainder of x when divided by 10 to s.
Step 6: Divide x by 10
Step 7: Repeat steps 4-7 till the condition is false.

Step 8: Print the calculated sum as the result.
Step 9: Stop

PROGRAM CODE:

```
#!/bin/bash
echo "Enter the num: "
read x
(( s = 0 ))
while [ $x -gt 0 ]
do
        (( s = s + x%10 ))
        (( x = x/10))
done
echo "Sum: $s"
~
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$ bash q4.sh
Enter the num:
565
Sum: 16
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$ bash q4.sh
Enter the num:
189
Sum: 18
ddcl36@ddcl36-Veriton-Series:~/Desktop/3524/bash$
```

RESULT:
Hence, a shell script has been written to find the sum of digits of an entered integer.

5. Find whether the given two array elements are the same or not. [Note: All the corresponding elements must be the Same]

AIM: To find whether the given two array elements are the same or not.

ALGORITHM:
Step 1: Start
Step 2: Read the size of the first array in a variable n1.
Step 3: Loop a variable from 0 to n1 and read values for the first array.
Step 4: Read the size of the second array in a variable n2.
Step 5: Loop a variable from 0 to n2 and read values for the second array.
Step 6: If n1 and n2 are not equal, skip to step 10 after printing that the arrays are not identical.

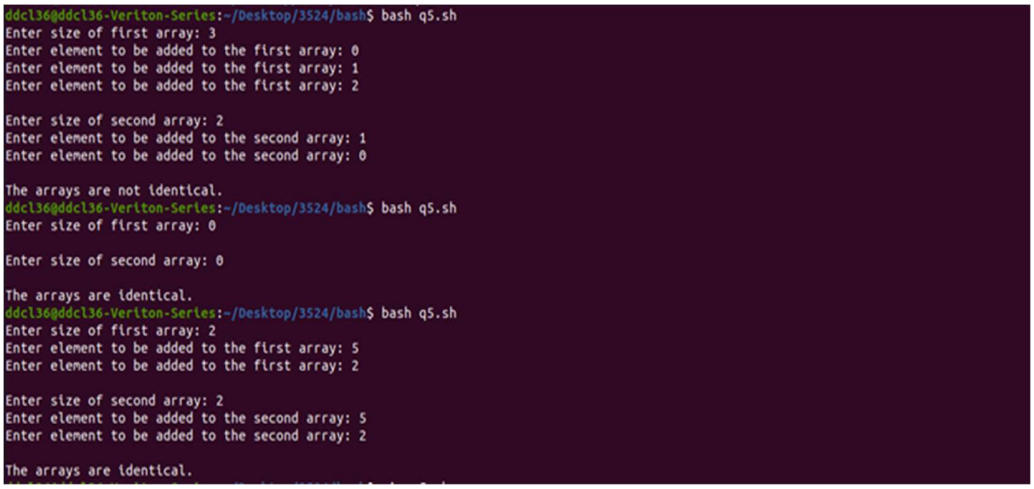
Step 7: Loop a variable from 0 to n1, and check whether the corresponding elements of the array are the same or not. If they are all found to be equal, go to step 8. Otherwise, skip to step 9.
Step 8: Print that they are identical and skip to step 10.
Step 9: Print that they are not identical and skip to step 10.
Step 10: Stop

PROGRAM CODE:

```
#!/bin/bash

arr1=()
arr2=()
echo -n "Enter size of first array: "
read n1

for (( i=n1; i>0; i-- ))
do
        #(( n1 -= 1 ))
        echo -n "Enter element to be added to the first array: "
        read x
        arr1+=($x)
done
#echo ${arr1[3]}

echo ""
echo -n "Enter size of second array: "
read n2
```

```
for (( i=n2; i>0; i-- ))
do
        echo -n "Enter element to be added to the second array: "
        read x
        arr2+=($x)
done
echo ""
if [ $n1 -ne $n2 ];
then
        echo "The arrays are not identical."
        exit
fi
for (( i=0; i<n1; i++ ))
do
        if [ ${arr1[i]} -ne ${arr2[i]} ];
        then
                echo "The arrays are not identical."
                exit
        fi
done
echo "The arrays are identical."
```

OUTPUT:



RESULT:

Hence, a shell script has been written to compare two given arrays and check if they are identical or not.

6. Write a shell program to create an array of binary equivalents for the given array of integers.

AIM: To write a shell program to create an array of binary equivalents for a given array of integers.

ALGORITHM:

Step 1: Start
Step 2: Write a function bina() to convert a given integer to its binary equivalent.
Step 3: For the conversion, collect the sequence of remainders upon dividing the integer by 2, and reverse the sequence.
Step 4: Read an input n for the size of the array from the user.
Step 5: Collect n elements from the user and append to the array.
Step 6: Loop the elements of the array and pass each value through the function bina().
Step 7: Append the binary equivalents returned onto another array binar.
Step 8: Loop through the values of binar, and print them.
Step 9: Stop

CODE:

```
#!/bin/bash
function bina ()
{
        (( arg = $1 ))
        (( bin = 0 ))
        #(( binfin = "" ))
        (( c = 0 ))
        while [ $arg -gt 0 ]
        do
                (( cond = arg % 2 ))
                (( bin = bin*10 + cond ))
                if [[ bin -eq 0 ]] && [[ cond -eq 0 ]];
                then
                        (( c = c+1 ))
                fi
                #bin += 10*bin + cond
                (( arg = arg/2 ))
                #echo "$cond"
        done
        #for (( i=0; i<c; i++ ))
        #do
        #       binfin+="${bin[i]}"
        #done
        (( bin = $bin * (10**$c) ))
        echo $bin
}
```

```
echo -n "Enter size of array: "
read n
arr=()
binar=()
for (( i=0; i<n; i++ ))
do
        echo -n "Enter element #$i: "
        read s
        arr+=($s)
done

echo "The binary values of entered numbers respectively: "

for (( i=0; i<n; i++ ))
do
        (( temp = ${arr[i]} ))
        (( temp = $(bina $temp) ))
        binar+=($temp)
done

for (( i=0; i<n; i++ ))
do
        echo "${binar[i]}"
done
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q6.sh
Enter size of array: 5
Enter element #0: 1
Enter element #1: 2
Enter element #2: 3
Enter element #3: 4
Enter element #4: 5
The binary values of entered numbers respectively:
1
10
11
100
101
```

RESULT:

Hence, a shell script has been written to find the binary equivalents of a given array of integers.

7. Write a shell program using a simple calculator function. Read an integer followed operator. If the operator is +,-,*,/ then read the next operand and compute the result. If the operator is = | then print the result so far accumulated. Perform integer type operations.

AIM:

To write a shell program using a simple calculator function by reading an integer followed by an operator till the '|' symbol is found.

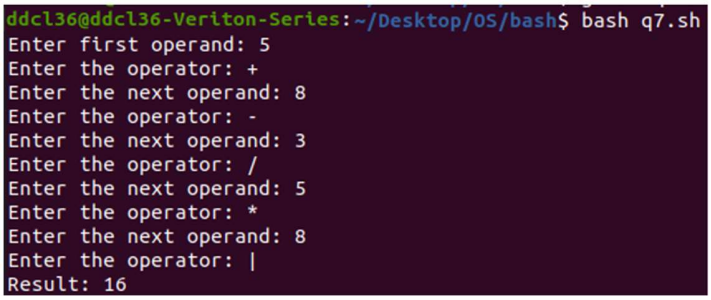
ALGORITHM:

Step 1: Start
Step 2: Read the first operand and store it in a variable n1.
Step 3: Create an infinite loop that asks for the following operator.
Step 4: If the operator is in (+, -, /, *), ask the user for the next operand, store it in n2 and skip to step 6.
Step 5: If the operator is '|', print the result accumulated so far and skip to step 7.
Step 6: Calculate the result, and go back to step 4.
Step 7: Stop

PROGRAM CODE:

```
#!/bin/bash
echo -n "Enter first operand: "
read n1

while [ 3 -gt 2 ]
do
        echo -n "Enter the operator: "
        read O
        case $O in
        "+")
                echo -n "Enter the next operand: "
                read n2
                (( res = n1 + n2 ));;
        "-")
                echo -n "Enter the next operand: "
                read n2
                (( res = n1 - n2 ));;
        "*")
                echo -n "Enter the next operand: "
                read n2
                (( res = n1 * n2 ));;
        "/")
                echo -n "Enter the next operand: "
                read n2
                (( res = n1 / n2 ));;
        "|")
                echo "Result: $res"
                exit;;
        esac
        (( n1 = res ))
done
~
```

OUTPUT:



RESULT:

Hence, a shell script has been written using a simple calculator function by reading an integer followed by an operator till the '|' symbol is found.

8. Write a shell program using functions to find if the given string is palindrome or not.

AIM:

Create an array of strings in the main function and display the string array which are palindromes.

ALGORITHM:

Step 1: Start
Step 2: Read the sentence
Step 3: create a function that accepts each string in sentence and loop for half of the length of the string
Step 4: Create 2 pointers one at front and other at end
Step 5: In each iteration check the two pointer if they point to same character
Step 6: If step 6 is true then increment front pointer by 1 and decrement end pointer by 1
Step 7: Return 1 if all the characters are equal else return 0
Step 8: call the function in main program and print it if string is palindrome
Step 9: Stop

PROGRAM CODE:

```
#!/bin/bash
function pal ()
{
        str=$1
        a=1
        len=$(expr length $str)
        for (( i=0; i<len/2; i++ ))
        do
                p=len-i-1
                if [ ${str:$i:1} != ${str:$p:1} ];
                then
                        a=0
                fi
        done
        echo $a
}
echo -n "Enter sentence: "
read sen
arr=()
len=0
for i in $sen;
do
        if [[ $(pal $i) -eq 1 ]];
        then
                arr+=($i)
                len=$len+1
        fi
done
echo "List of palindromic strings found: "
for (( i=0; i<len; i++ ))
do
        echo "${arr[$i]}"
done
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q8.sh
Enter sentence: my dad drives a racecar under the radar
List of palindromic strings found:
dad
a
racecar
radar
mithrank@mithrank-VirtualBox:~/Desktop/OS/bash$
```

RESULT:

Hence, a shell script has been written using functions to find if the given string is palindrome or not.

9. Write a shell program to check whether the given integer is palindrome.

AIM:

To write a shell program to check whether the given integer is palindrome.

ALGORITHM:

Step 1: Start
Step 2: Read the integer
Step 3: create a function that accepts integer and loop for half of the length of the number
Step 4:Create 2 pointers one at front and other at end
Step 5: In each iteration check the two pointer if they point to same number
Step 6: If step 6 is true then increment front pointer by 1 and decrement end pointer by 1
Step 7: return 1 if all the numbers are equal else return 0
Step 8: call the function in main program and print it if number is palindrome
Step 9: Stop

CODE:

```
#!/bin/bash
function pal ()
{
        str=$1
        a=1
        len=$(expr length $str)
        for (( i=0; i<len/2; i++ ))
        do
                p=len-i-1
                if [ ${str:$i:1} != ${str:$p:1} ];
                then
                        a=0
                fi
        done
        echo $a
}

echo -n "Enter num: "
read n

if [[ $(pal $n) -eq 1 ]];
then
        echo "Entered num is a panlindrome."
else
        echo "Entered num is not a palindrome."
fi
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q9.sh
Enter num: 12345
Entered num is not a palindrome.
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q9.sh
Enter num: 343
Entered num is a panlindrome.
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q9.sh
Enter num: 4
Entered num is a panlindrome.
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$
```

RESULT:

Hence, a shell script has been written to check whether the given integer is palindrome.

10. The students of a class are sitting in M rows with each row consisting of N seats. The students should sit in roll number order from row 1(occupying first N students) to row M. If a student is absent then the seat is kept vacant. Now you are given a binary matrix of order MXN, where 1 denotes the seat occupied by the student for the seat and 0 denotes that student is absent print the roll numbers of the absentees as per the given binary matrix.

AIM:

To write a shell script to print the roll numbers of the absentees as per the given binary matrix.

ALGORITHM:

Step 1: Start
Step 2: Read the number of rows and columns.
Step 3: loop through each cell and enter a random number mod 2 in each cell
Step 4: Again loop through each cell and make count of number of cells having zeros
Step 5: Calculate absentees a using the formula (no_of_rows*current_row_no + current_column_no + 1)
Step 6: Display absentees in each loop
Step 7: Stop

CODE:

```
#!/bin/bash
echo -n "Enter no. of rows (M): "
read M
echo -n "Enter no. of columns (N): "
read N
mat=()
for (( i=0; i<M; i++ ))
do
        for (( j=0; j<N; j++ ))
        do
                (( tmp = $RANDOM % 2 ))
                mat+=($tmp)
                echo -n "$tmp "
        done
        echo ""
done
c=0
echo "Roll numbers of absentees: "
for (( i=0; i<M; i++ ))
do
        for (( j=0; j<N; j++ ))
        do
                if [[ ${mat[$i*$N+$j]} -eq 0 ]];
                then
                        echo $c
                fi
                (( c = c+1 ))
        done
done
~
~
```

OUTPUT:

```
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q10.sh
Enter no. of rows (M): 3
Enter no. of columns (N): 6
0 1 0 1 0 1
1 0 1 1 1 1
0 0 0 1 1 0
Roll numbers of absentees:
0
2
4
7
12
13
14
17
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$ bash q10.sh
Enter no. of rows (M): 2
Enter no. of columns (N): 4
1 0 1 1
0 1 0 1
Roll numbers of absentees:
1
4
6
ddcl36@ddcl36-Veriton-Series:~/Desktop/OS/bash$
```

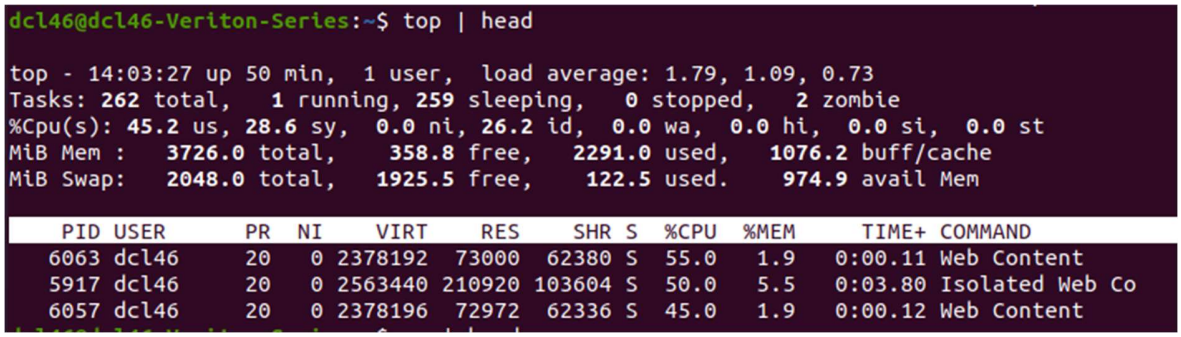
RESULT:

Thus, a shell script was written for printing the roll numbers of the absentees as per the given binary matrix.

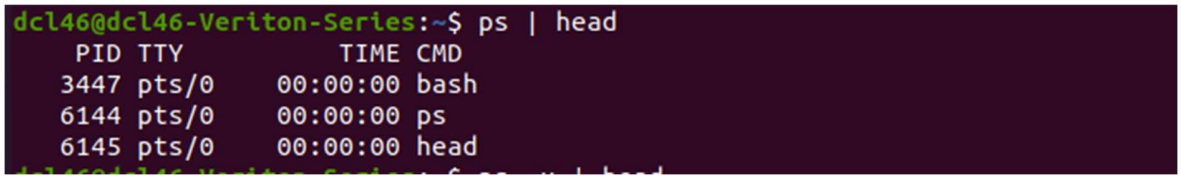
EXP NO: 4 DATE : 8-4-2022	Process Commands

AIM: To execute the process commands in LINUX.

1.$top : displays all running process

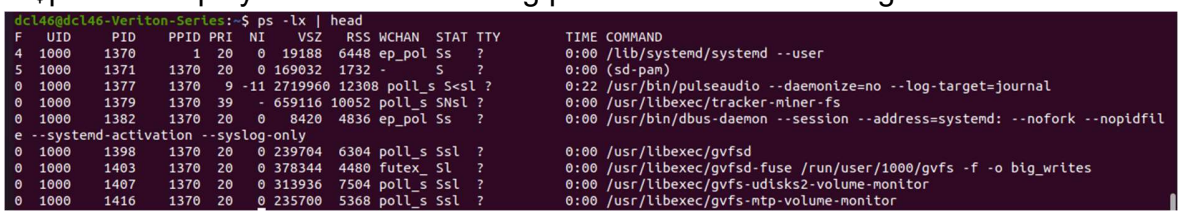


2.$ps : displays the current running process



3.$ps -u : selects the processes whose effective user name or ID is in userlist.

4.$ps -lx : displays the current running process in extended long format



5.$pstree : displays the running processes as tree

```
dcl46@dcl46-Veriton-Series:~$ pstree | head
systemd-+-ModemManager---2*[{ModemManager}]
        |-NetworkManager---2*[{NetworkManager}]
        |-accounts-daemon---2*[{accounts-daemon}]
        |-acpid
        |-avahi-daemon---avahi-daemon
        |-colord---2*[{colord}]
        |-cron
        |-cups-browsed---2*[{cups-browsed}]
        |-cupsd
        |-dbus-daemon
```

6.$ps -A : Selects all the processes

```
dcl46@dcl46-Veriton-Series:~$ ps -A | head
    PID TTY          TIME CMD
      1 ?        00:00:01 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-events_highpri
      8 ?        00:00:00 mm_percpu_wq
      9 ?        00:00:00 rcu_tasks_rude_
     10 ?        00:00:00 rcu_tasks_trace
     11 ?        00:00:00 ksoftirqd/0
```

7.$ps -au :

```
dcl46@dcl46-Veriton-Series:~$ ps -au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
dcl46       1457  0.0  0.1 164016  5788 tty2     Ssl+ 13:19   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubunt
dcl46       1459  1.8  1.3 381008 52816 tty2     Rl+  13:19   2:18 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -ba
dcl46       1477  0.0  0.3 188088 12288 tty2     Sl+  13:19   0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu
dcl46       3447  0.0  0.1  10748  4520 pts/0    Ss   13:22   0:00 bash
dcl46       9786  0.0  0.0  11496  3276 pts/0    R+   15:23   0:00 ps -au
```

8.$kill Process_id : killing a process with the mentioned process id

```
 3643 dcl46     20   0 3348708 423172 177980 S   8.6  11.1   0:36.52 firefox
```

```
dcl46@dcl46-Veriton-Series:~$ kill 3643
```

9.$kill -l : Lists all types of kill commands

```
dcl46@dcl46-Veriton-Series:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

10.$kill -9 Process_id : forced killing of a process with the mentioned process id

Top command to obtain PID of process to kill

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  10218 dcl46     20   0 3181928 324212 153908 R  70.6   8.5   0:05.50 firefox
```

Killing the process

```
dcl46@dcl46-Veriton-Series:~$ kill -9 10218
dcl46@dcl46-Veriton-Series:~$
```

11.$nice -value -gnome-terminal app_name

```
dcl46@dcl46-Veriton-Series:~$ nice -15 gnome-terminal
dcl46@dcl46-Veriton-Series:~$ nice -15 gnome-terminal chrome
dcl46@dcl46-Veriton-Series:~$
```

RESULT:

Hence the above process commands are executed in LINUX and the output is verified.

EXP NO: 5 DATE : 8-4-2022	PROCESS MANAGEMENT

1)Write a C program to fork two separate processes.

AIM:

To write a C program to fork two separate processes.

PROGRAM CODE:

```
#include <stdio.h>
#include <unistd.h>
void main()
{
  int pi_d ;
  int pid ;
  pi_d = fork(); //create new process
  if(pi_d == 0)  {
    printf("Child Process A:\nPID: %d\nPPID: %d\n",getpid(),getppid());
  }
  if(pi_d > 0)  {
    pid = fork();
    if(pid > 0)  {
      printf("\nParent Process:\nPID:%d\nPPID: %d\n",getpid(),getppid());
    }
    else if(pid == 0)  {
      printf("Child Process B:\nPID: %d\nPPID: %d\n",getpid(),getppid());
    }
  }
}
```

OUTPUT:

```
Parent Process:
PID:4651
PPID: 2072
```

CHILD PROCESS

```
PID: 4653
PPID: 1293
Child Process A:
PID: 4652
PPID: 1293
```

RESULT:

Hence the program is executed and the output is verified.

2)Write a program which creates a child process, waits for a termination of its child and lists its PID, together with the state in which the process was terminated.

AIM:

To write a program which creates a child process, waits for a termination of its child and lists its PID, together with the state in which the process was terminated.

PROGRAM CODE:

```
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
int main()
{
int pid, state;
printf("\nBefore the fork :\n");
if ((pid =fork() != 0))
{
  printf("Parent: \n");
  wait(&state);
}
else
{
  printf("Child:\n");
  execl("./child", "child",NULL);
  perror("Error execution");
}
printf("After fork state = %d\n",state);
printf("PID child = %d Terminated\n", pid);
return 0;
}
```

OUTPUT:

```
Before the fork :
Parent:
Child:
Error execution: No such file or directory
After fork state = 1833164608
PID child = 0 Terminated
After fork state = 0
PID child = 1 Terminated
```

RESULT:

Hence the program is executed and the output is verified.

3)To write a C program to create an orphan & zombie process.

AIM:

To write a C program to create an orphan & zombie process.

CODE:

1. **Zombie process**

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        pid_t child_pid = fork();
        if (child_pid > 0)
                sleep(50);
        else
                exit(0);

        return 0;
}
```

OUTPUT:

```
~/Desktop/processcommands$ cc 3a.c
~/Desktop/processcommands$ ./a.out
```

2. Orphan process

```
#include<stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        int pid = fork();
        if (pid > 0)
                printf("in parent process");
        else if (pid == 0)
        {
                sleep(30);
                printf("in child process");
        }
        return 0;
}
```

OUTPUT:

```
~/Desktop/processcommands$ cc 3b.c
~/Desktop/processcommands$ ./a.out
in parent process
```

RESULT:

Hence the program is executed and the output is verified.

4)Write a C program which shows how two processes which are members of the parent-child relationship are concurrent from the execution point of view, initially the child is a copy of the parent but every process has its own data.

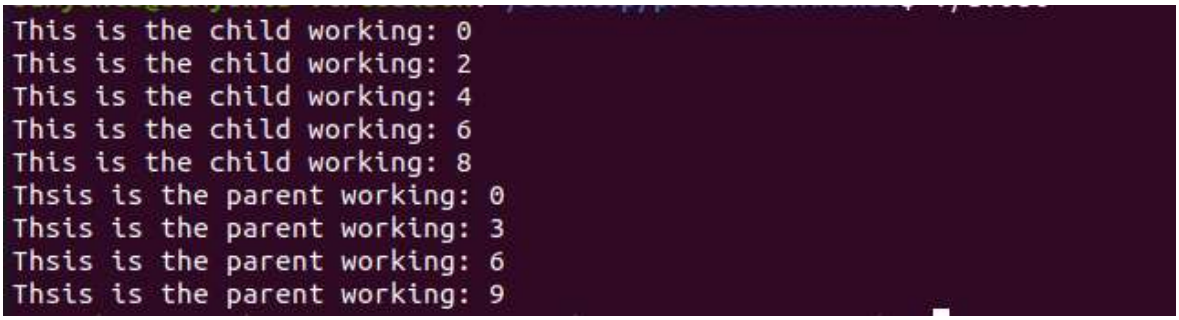
AIM:

Write a C program which shows how two processes which are members of the relationship parent-child are concurrent from the execution point of view, initially the child is a copy of the parent but every process has its own data.

CODE:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main( void)
{
int pid;
int n=0;
int status;
pid= fork();
  while (n<10)
  {
    if (pid == 0)
    {
        printf("This is the child working: %d\n", n);
        n = n + 2;
        sleep(1);
    }
    else
    {
        wait(&status);
        printf("This is is the parent working: %d\n", n);
        n = n +3;
        sleep(1);
    }
  }
}
```

OUTPUT:



RESULT:
Hence the program is executed and the output is verified.

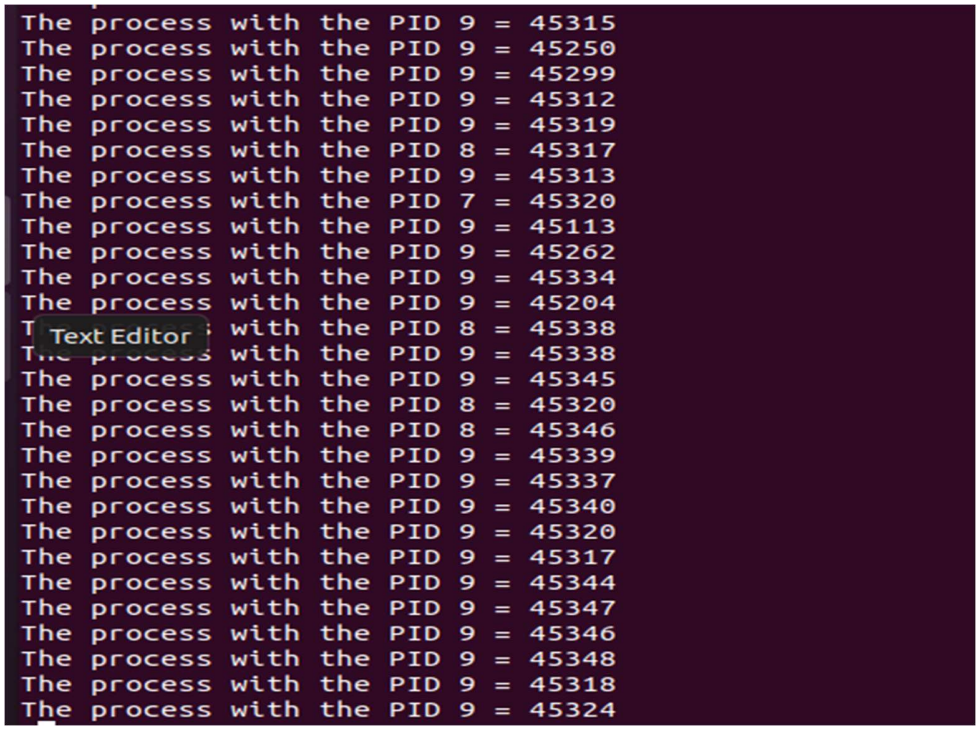
5)Test the source code below:

AIM:
To test the source code given and produce the output.

CODE:

```
#include <stdio.h>
#include <unistd.h>
int main()
{
int pid ;
for(int i=1;i<=10;i++)
{

   pid = fork();
   if(pid==0)
   printf("The process with the PID %d = %d\n",i,getpid());
}
return 0;
}
```

OUTPUT:



We get 2^10 processes generated as a result of this program due to 10 forks being called in the for loop.

RESULT:

Hence the program is executed and the output is verified.

EXP NO: 6 DATE : 10-5-2022	THREAD PROGRAMMING

AIM:

i) To perform basic functions on pthreads.

ii)To display multi threads with global and static variables.

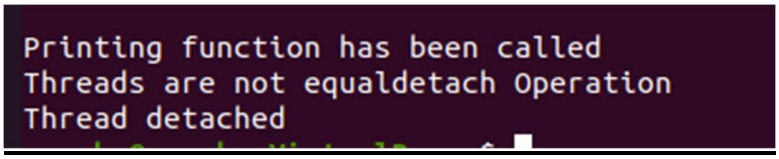
1)BASIC FUNCTIONS ON PTHREADS

CODE:

```
#include <pthread.h>
#include <stdio.h>
void *printmsg(){
        printf("\nPrinting function has been called");
}
int threadcompare(){
        pthread_t t1;
        pthread_create(&t1,NULL,&printmsg,NULL);
        return (pthread_equal(t1,pthread_self()));
}
void threadfunc(){
        pthread_t thread1;
        pthread_create(&thread1,NULL,&printmsg,NULL);
        pthread_join(thread1,(void *)pthread_self());
}
void thread_detach(){
        pthread_t thread_2;
        pthread_create(&thread_2 , NULL , &printmsg , NULL);
        printf("detach Operation");
        pthread_detach(pthread_self());
}
int main(){
        threadfunc();
        if (threadcompare()) printf("\nThreads are equal");
        else printf("\nThreads are not equal");
        thread_detach();
        printf("\nThread detached\n");
```

```
        return 0;
}
```

OUTPUT:



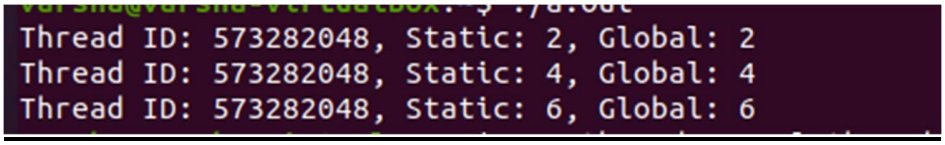
2)DISPLAY GLOBAL AND STATIC VARIABLES IN MULTI THREADS

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
        // Store the value argument passed to this thread
        int *myid = (int *)vargp;

        // Let us create a static variable to observe its changes
        static int s = 0;

        // Change static and global variables
        ++s; ++g;

        // Print the argument, static and global variables
        printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
        int i;
```

```
        pthread_t tid;

        // Let us create three threads
        for (i = 0; i < 3; i++)
                pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

        pthread_exit(NULL);
        return 0;
}
```

OUTPUT:



RESULT:

Thus, the pthread functions and the multi threads with global variables and static variables were displayed and output was verified.

EXP NO: 7 DATE : 13-5-2022	CPU SCHEDULING ALGORITHMS

AIM:

To implement all the CPU Process scheduling Algorithms.

(i)FCFS ALGORITHM

ALGORITHM:

i. Declare an array for Burst time.
ii. Get user input for Burst time.
iii. Create a new array for calculating Completion time, Turn Around time and Wait time.
iv. Run a for loop that calculates,
CT[i] = CT[i-1] + BT[i]
TAT[i] = CT[i] – i (Arrival time)
WT[i] = TAT[i] – BT[i]
v. Print the result accordingly.

PROGRAM CODE:

```
#include<iostream>
using namespace std;
int main()
{
        int *BT,*CT,*TAT,*WT,n;
        float avgTAT = 0.0,avgWT = 0.0;
        cout<<"First Come First Serve Algorithm.";
        cout<<"\n Enter the number of processes: ";
        cin>>n;
        BT = new int[n+1];
        CT = new int[n+1];
        TAT = new int[n+1];
        WT = new int[n+1];
```

```
	BT[0] = 0;
	TAT[0] = 0;
	CT[0] = 0;
	WT[0] = 0;
	for(int i=1;i<=n;i++)
	{
		cout<<"\nEnter Burst Time of Process "<<i<<": ";
		cin>>BT[i];
		CT[i] = CT[i-1] + BT[i];
		TAT[i] = CT[i] - (i-1);
		WT[i] = TAT[i] - BT[i];
		avgTAT = avgTAT + TAT[i];
		avgWT = avgWT + WT[i];
	}
	avgTAT = avgTAT/float(n);
	avgWT = avgWT/float(n);
	cout<<"\n P.NO\t AT\t BT\t CT\t TAT\t WT";
	for(int i = 1;i<=n;i++)
		cout<<"\n "<<i<<"\t "<<i-1<<"\t "<<BT[i]<<"\t "<<CT[i]<<"\t "<<TAT[i]<<"\t
"<<WT[i];
	cout<<"\n\n Average TAT: "<<avgTAT;
	cout<<"\n Average WT: "<<avgWT<<"\n";
	return 0;
}
```

OUTPUT:

```
First Come First Serve Algorithm.
 Enter the number of processes: 6

Enter Burst Time of Process 1: 4

Enter Burst Time of Process 2: 2

Enter Burst Time of Process 3: 1

Enter Burst Time of Process 4: 3

Enter Burst Time of Process 5: 5

Enter Burst Time of Process 6: 2

 P.NO    AT      BT      CT      TAT     WT
 1       0       4       4       4       0
 2       1       2       6       5       3
 3       2       1       7       5       4
 4       3       3       10      7       4
 5       4       5       15      11      6
 6       5       2       17      12      10

 Average TAT: 7.33333
 Average WT: 4.5
```

(ii) SJF ALGORITHM

ALGORITHM:

i. Initialize variable n with number of processes and get all the process' Burst time.
ii. Run process 1, that has arrived at arrival time 0 and add it into a check variable and make its BT = 0;
iii. Now, for every process that has arrival time less than the check variable, sort the process.
iv. Execute the shortest process and initialize its Burst time as infinity.
v. Until every process is completed, run the above 2 steps and calculate,
CT[i] = comp
TAT[i] = CT[i] – AT[i]
WT[i] = TAT[i] – BT
vi. Print the output accordingly.

PROGRAM CODE:

```
#include<bits/stdc++.h>
```

```
using namespace std;

void sortall(int *BT, int *PID, int *AT, int *CT, int *TAT, int *WT, int *tempBT, int n) {
for(int i=0;i<n;i++)
   {
      int pos=i;
      for(int j=i+1;j<n;j++)
      {
         if(BT[j]<BT[pos])
            pos=j;
      }
      int temp=BT[i];
      BT[i]=BT[pos];
      BT[pos]=temp;

      temp=PID[i];
      PID[i]=PID[pos];
      PID[pos]=temp;

      temp=AT[i];
      AT[i]=AT[pos];
      AT[pos]=temp;

      temp=TAT[i];
      TAT[i]=TAT[pos];
      TAT[pos]=temp;

      temp=CT[i];
      CT[i]=CT[pos];
      CT[pos]=temp;

      temp=WT[i];
      WT[i]=WT[pos];
      WT[pos]=temp;

      temp=tempBT[i];
      tempBT[i]=tempBT[pos];
      tempBT[pos]=temp;
   }
```

```
}

int main()
{
   int *BT,*CT,*TAT,*WT,*PID,*AT,*tempBT,n;
   float avgTAT = 0.0,avgWT = 0.0;
   cout<<"Shortest Job First Algorithm.";
   cout<<"\n Enter the number of processes: ";
   cin>>n;
   BT = new int[n];
   CT = new int[n];
   AT = new int[n];
   TAT = new int[n];
   WT = new int[n];
   PID = new int[n];
   tempBT = new int[n];
   for(int i=0;i<n;i++)
   {
      cout<<"\nEnter Burst Time of Process "<<i+1<<": ";
      cin>>BT[i];
      tempBT[i] = BT[i];
      cout<<"\n"<<BT[i];
      AT[i] = i;
      PID[i] = i+1;
   }
   int comp = 0;
   for(int i = 0;i<n;i++)
   {
      CT[0] = comp + tempBT[0];
      cout<<"\nCT : "<<CT[0];
      TAT[0] = CT[0] - AT[0];
            cout<<" TAT : "<<TAT[0];

      WT[0] = TAT[0] - tempBT[0];
            cout<<" WT : "<<WT[0];

      BT[0] = INT_MAX;
            cout<<" BT : "<<tempBT[0];

      comp = CT[0];
```

```
            cout<<" comp : "<<comp;
            cout<<" AT : "<<AT[0];

        if(comp<n)
            sortall(BT,PID,AT,CT,TAT,WT,tempBT,comp);
        else
            sortall(BT,PID,AT,CT,TAT,WT,tempBT,n);
    }
    cout<<"\n P.NO\t AT\t BT\t CT\t TAT\t WT";
    for(int i=0;i<n;i++)
    {
        avgTAT = avgTAT + TAT[i];
        avgWT = avgWT + WT[i];
        cout<<"\n "<<PID[i]<<"\t "<<AT[i]<<"\t "<<tempBT[i]<<"\t "<<CT[i]<<"\t
"<<TAT[i]<<"\t "<<WT[i];
    }
    avgTAT = avgTAT/float(n);
    avgWT = avgWT/float(n);
    cout<<"\n\n Average TAT: "<<avgTAT;
    cout<<"\n Average WT: "<<avgWT<<"\n";
    return 0;
}
```

OUTPUT:

```
Enter Burst Time of Process 1: 7

7
Enter Burst Time of Process 2: 5

5
Enter Burst Time of Process 3: 1

1
Enter Burst Time of Process 4: 2

2
Enter Burst Time of Process 5: 8

8
CT : 7 TAT : 7 WT : 0 BT : 7 comp : 7 AT : 0
CT : 8 TAT : 6 WT : 5 BT : 1 comp : 8 AT : 2
CT : 10 TAT : 7 WT : 5 BT : 2 comp : 10 AT : 3
CT : 15 TAT : 14 WT : 9 BT : 5 comp : 15 AT : 1
CT : 23 TAT : 19 WT : 11 BT : 8 comp : 23 AT : 4
 P.NO    AT      BT      CT      TAT     WT
 5       4       8       23      19      11
 2       1       5       15      14      9
 4       3       2       10      7       5
 3       2       1       8       6       5
 1       0       7       7       7       0

 Average TAT: 10.6
 Average WT: 6
```

(iii) Round Robin ALGORITHM.

ALGORITHM:

i. Initialize an integer queue, and n for number of processes.
ii. Get BT for every process and get time quantum.
iii. Run the first arrived process for the time quantum amount of time and subtract it from BT of that process and update comp.
iv. Add all the process ID of the process that has arrived within the comp var-iable.
v. Pop from the queue and run that process.
vi. Loop the above 3 steps until the queue is empty.
vii. Compute,
CT[i] = comp
TAT[i] = CT[i] – AT[i]
WT[i] = TAT[i] – BT
viii. Print the result accordingly.

PROGRAM CODE:

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

typedef struct Queue
{
	int val[200];
	int front;
	int rear;
}queue;

queue createQueue()
{
	queue q;
	q.front = 0;
	q.rear = 0;
	return q;
}

int isEmpty(queue q)
{
	if(q.front == q.rear)
		return 1;
	return 0;
}

int isFull(queue q)
{
	if((q.rear+1)%200 == q.front)
		return 1;
	return 0;
}

int insert(queue *q, int data)
{
	if(isFull(*q))
		return 0;
	q->rear = (q->rear + 1)%200;
	q->val[q->rear] = data;
```

```
        return 1;
}

int del(queue *q, int *data)
{
        if(isEmpty(*q))
                return 0;
        q->front = (q->front + 1)%200;
        *data = q->val[q->front];
        return 1;
}

int main()
{
        int *BT,*CT,*TAT,*WT,*PID,*AT,*tempBT,*visited,n,tq;
        float avgTAT = 0.0,avgWT = 0.0;
        cout<<"Round Robin First Algorithm.";
        cout<<"\n Enter the number of processes: ";
        cin>>n;
        BT = new int[n];
        CT = new int[n];
        AT = new int[n];
        TAT = new int[n];
        WT = new int[n];
        PID = new int[n];
        tempBT = new int[n];
        visited = new int[n];
        cout<<"\n Enter time quantum: ";
        cin>>tq;
        for(int i=0;i<n;i++)
        {
                cout<<"\nEnter Burst Time of Process "<<i+1<<": ";
                cin>>BT[i];
                tempBT[i] = BT[i];
                AT[i] = i;
                PID[i] = i+1;
                visited[i] = 0;
        }
        int comp = 0,p;
        queue q = createQueue();
```

```
        insert(&q, 0);
        visited[0]=1;
        do
        {
                del(&q,&p);
                if(BT[p] > tq)
                {
                        comp = comp + tq;
                        BT[p] = BT[p] - tq;
                }
                else
                {
                        comp = comp + BT[p];
                        BT[p] = 0;
                        CT[p] = comp;
                        //cout<<"\n Comp of "<<p<<":"<<CT[p];
                }
                for(int i = 0; i<n ; i++)
                {
                        if((AT[i]<=comp)&&(!visited[i]))
         {
            insert(&q, i);
            visited[i]=1;
            //cout<<"\n After "<<p<<" inserting "<<i;
         }
                }
                if(BT[p])
                        insert(&q, p);
        }while(!isEmpty(q));

        cout<<"\n P.NO\t AT\t BT\t CT\t TAT\t WT";
        for(int i=0;i<n;i++)
        {
                TAT[i] = CT[i] - AT[i];
                WT[i] = TAT[i] - tempBT[i];
                avgTAT = avgTAT + TAT[i];
                avgWT = avgWT + WT[i];
                cout<<"\n "<<PID[i]<<"\t "<<AT[i]<<"\t "<<tempBT[i]<<"\t "<<CT[i]<<"\t
"<<TAT[i]<<"\t "<<WT[i];
        }
```

```
        avgTAT = avgTAT/float(n);
        avgWT = avgWT/float(n);
        cout<<"\n\n Average TAT: "<<avgTAT;
        cout<<"\n Average WT: "<<avgWT<<"\n";
        return 0;
}
```

OUTPUT:

```
 Enter the number of processes: 6

 Enter time quantum: 2

Enter Burst Time of Process 1: 4

Enter Burst Time of Process 2: 5

Enter Burst Time of Process 3: 2

Enter Burst Time of Process 4: 1

Enter Burst Time of Process 5: 3

Enter Burst Time of Process 6: 6

 P.NO    AT      BT      CT      TAT     WT
 1       0       4       8       8       4
 2       1       5       17      16      11
 3       2       2       6       4       2
 4       3       1       9       6       5
 5       4       3       16      12      9
 6       5       6       21      16      10

 Average TAT: 10.3333
 Average WT: 6.83333
```

(iv) PRIORITY SCHEDULING ALGORITHM

ALGORITHM:

1. Input the processes with their arrival time, burst time and priority.
2. First process will schedule, which has the lowest arrival time, if two or more processes will have the lowest arrival time, then whoever has higher priority will schedule first.
3. Now further processes will be scheduled according to the arrival time and priority of the process. (Here we are assuming that lower priority numbers have higher priority). If two process priorities are the same then sort according to process number.
4. Once all the processes have been arrived, we can schedule them based on their priority.

PROGRAM CODE:

```
#include<iostream>
using namespace std;
int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   cout<<"Enter Total Number of Process:";
   cin>>n;

   cout<<"\nEnter Burst Time and Priority\n";
   for(i=0;i<n;i++)
   {
      cout<<"\nP["<<i+1<<"]\n";
      cout<<"Burst Time:";
      cin>>bt[i];
      cout<<"Priority:";
      cin>>pr[i];
      p[i]=i+1;
   }

   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(pr[j]<pr[pos])
            pos=j;
      }
      temp=pr[i];
      pr[i]=pr[pos];
      pr[pos]=temp;
      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;
      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }
```

```
    wt[0]=0;
    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=total/n;    total=0;

    cout<<"\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time";
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        cout<<"\nP["<<p[i]<<"]\t\t  "<<bt[i]<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];
    }
    avg_tat=total/n;     //average turnaround time
    cout<<"\n\nAverage Waiting Time="<<avg_wt;
    cout<<"\nAverage Turnaround Time="<<avg_tat;

    return 0;
}
```

OUTPUT:

```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:10
Priority:1

P[2]
Burst Time:20
Priority:3

P[3]
Burst Time:40
Priority:2

P[4]
Burst Time:20
Priority:1

Process     Burst Time          Waiting Time    Turnaround Time
P[1]              10                0                 10
P[4]              20                10                30
P[3]              40                30                70
P[2]              20                70                90

Average Waiting Time=27
Average Turnaround Time=50
```

RESULT: Hence all CPU Scheduling Algorithms implemented successfully.

EXP NO: 6 DATE : 10-5-22	INTRODUCTION TO xv6: DOWNLOAD AND BUILD (i) SCHEDULING IN xv6

AIM: To download and build the xv6 and implement scheduling in xv6

STEP 1:

sudo apt-get install qemu

```
rusa@rusa-Veriton-M4660G:~$ sudo apt-get install qemu
[sudo] password for rusa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
qemu is already the newest version (1:4.2-3ubuntu6.21).
0 upgraded, 0 newly installed, 0 to remove and 298 not upgraded.
```

STEP 2: sudo apt install git

```
rusa@rusa-Veriton-M4660G:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.4).
0 upgraded, 0 newly installed, 0 to remove and 298 not upgraded.
```

STEP 3: git clone https://github.com/mit-pdos/xv6-public

```
rusa@rusa-Veriton-M4660G:~$ git clone https://github.com/mit-pdos/xv6-public
fatal: destination path 'xv6-public' already exists and is not an empty directo
ry.
```

STEP 4: cd xv6-public

```
rusa@rusa-Veriton-M4660G:~$ cd xv6-public
```

STEP 5: sudo apt-get install build-essential

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 298 not upgraded.
```

STEP 6: sudo apt-get install gcc-multilib

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo apt-get install gcc-multilib
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc-multilib is already the newest version (4:9.3.0-1ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 298 not upgraded.
```

STEP 7: make

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ make
make: 'xv6.img' is up to date.
```

STEP 8: make qemu-nox

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ make qemu-nox
qemu-system-x86_64 -nographic -drive file=fs.img,index=1,media=disk,format=raw
-drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
make: qemu-system-x86_64: Command not found
make: *** [Makefile:231: qemu-nox] Error 127
```

After this we will get error 127

STEP 9: change 54 line in makefile [NOTE: STEP 9 is for 64-bit users]

QEMU = qemu-system-x86_64

```
53 # If the makefile can't find QEMU, specify its path here
54 QEMU = qemu-system-x86_64
55 # QEMU = qemu-system-i386
```

STEP 10: after step 9 again type make qemu-nox

We will get error 127

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ make qemu-nox
qemu-system-x86_64 -nographic -drive file=fs.img,index=1,media=disk,format=raw
-drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
make: qemu-system-x86_64: Command not found
make: *** [Makefile:231: qemu-nox] Error 127
```

STEP 11: grep -Eoc '(vmx|svm)' /proc/cpuinfo

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ grep -Eoc '(vmx|svm)' /proc/cpuinfo
16
```

STEP 12: sudo apt install qemu-kvm libvirt-daemon-system libvirt- clients bridge-utils virtinst virt-manager

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo apt install qemu-kvm libvirt-daemon
-system libvirt-clients bridge-utils virtinst virt-manager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpu-checker dmeventd gir1.2-appindicator3-0.1 gir1.2-gtk-vnc-2.0
  gir1.2-libosinfo-1.0 gir1.2-libvirt-glib-1.0 gir1.2-spiceclientglib-2.0
  gir1.2-spiceclientgtk-3.0 i965-va-driver ibverbs-providers
  intel-media-va-driver ipxe-qemu ipxe-qemu-256k-compat-efi-roms libaio1
  libcacard0 libdevmapper-event1.02.1 libfdt1 libgovirt-common libgovirt2
  libgtk-vnc-2.0-0 libgvnc-1.0-0 libibverbs1 libigdgmm11 libiscsi7
  liblvm2cmd2.03 libnss-mymachines libnss-systemd libosinfo-1.0-0
```

STEP 13: sudo systemctl is-active libvirtd

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo systemctl is-active libvirtd
active
```

STEP 14: sudo usermod -aG libvirt $USER

STEP 15: sudo usermod -aG kvm $USER

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo usermod -aG libvirt $USER
rusa@rusa-Veriton-M4660G:~/xv6-public$ sudo usermod -aG kvm $USER
rusa@rusa-Veriton-M4660G:~/xv6-public$ ls
asm.h             forktest.asm  lapic.d      proc.o        sysproc.o
bio.c             forktest.c    lapic.o      pr.pl         toc.ftr
bio.d             forktest.d    LICENSE      README        toc.hdr
bio.o             forktest.o    _ln          _rm           trapasm.o
bootasm.d         fs.c          ln.asm       rm.asm        trapasm.S
bootasm.o         fs.d          ln.c         rm.c          trap.c
bootasm.S         fs.h          ln.d         rm.d          trap.d
bootblock         fs.img        ln.o         rm.o          trap.o
bootblock.asm     fs.o          ln.sym       rm.sym        traps.h
bootblock.o       gdbutil       log.c        runoff        TRICKS
bootblockother.o  _grep         log.d        runoff1       types.h
bootmain.c        grep.asm      log.o        runoff.list   uart.c
bootmain.d        grep.c        _ls          runoff.spec   uart.d
bootmain.o        grep.d        ls.asm       _sh           uart.o
buf.h             grep.o        ls.c         sh.asm        ulib.c
BUGS              grep.sym      ls.d         sh.c          ulib.d
_cat              ide.c         ls.o         sh.d          ulib.o
cat.asm           ide.d         ls.sym       sh.o          umalloc.c
```

STEP 16: After opening of terminal give ls

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ make qemu SCHEDFLAG=DEFAULT
qemu-system-x86_64 -serial mon:stdio -drive file=fs.img,index=1,media=disk,form
at=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta
rt 58
init: starting sh
```

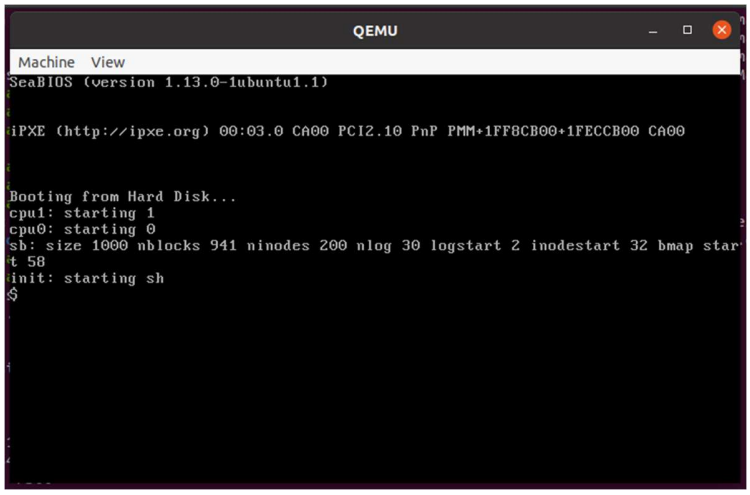
STEP 17: For terminating give ctrl+a. Then press x.

Scheduling in XV6:

1. Default – Round Robin scheduling

$ make qemu SCHEDFLAG=DEFAULT

```
rusa@rusa-Veriton-M4660G:~/xv6-public$ make qemu SCHEDFLAG=DEFAULT
qemu-system-x86_64 -serial mon:stdio -drive file=fs.img,index=1,media=disk,form
at=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta
rt 58
init: starting sh
```



For terminating, give ctrl+a. Then press x.

RESULT:

Hence downloaded and built the xv6 and implemented scheduling in xv6 and the output is verified.

EXP NO: 8 DATE : 27-05-2022	FAMILIARIZATION WITH DISK SCHEDULING ALGORITHMS

(I) First Come First Serve (FCFS) ALGORITHM:

AIM:

To implement the First Come First Serve (FCFS) Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start
2. Get the user input of requests to seek locations and store it in an array in ascending order of their time of arrival.
3. The position of the disk head is represented by 'Head'.
4. One by one Take the requested tracks indefaultorderand calculate absolute distance of it from the head.
5. Increment the total seek count with this distance.
6. Recently updated track position now becomes the new head
7. Repeat steps 2-5 until all requested tracks have been seeked.
8. Stop

PROGRAM CODE:

```
‘#include<stdio.h>
#include<stdlib.h>
int main()
{
        int RQ[100],i,n,TotalHeadMoment=0,initial;
        printf("Enter the number of Requests\n");
        scanf("%d",&n);
        printf("Enter the Requests sequence\n");
        for(i=0;i<n;i++)
        scanf("%d",&RQ[i]);
        printf("Enter initial head position\n");
        scanf("%d",&initial);

        // logic for FCFS disk scheduling
```

```
        for(i=0;i<n;i++)
        {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
        }

        printf("Total head moment is %d",TotalHeadMoment);
        return 0;
}
```

OUTPUT:

RESULT:

Hence, First Come First Serve (FCFS) Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

(II) Shortest Seek Time First (SSTF) Algorithm:

AIM:

To implement the Shortest Seek Time First (SSTF) Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start.
2. Let Request array represents an array storing indexes of tracks that have been requested. ‘head’ is the position of disk head.
3. Find the positive distance of all tracks in the request array from head.
4. Find a track from requested array which has not been accessed/serviced yet and has minimum distance from head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 2 until all tracks in request array have not been serviced.
8. Stop.

PROGRAM CODE:

```
#include<bits/stdc++.h>
using namespace std;
void sstfalgo(vector<int> req_arr,int head,int n)
{
        int count = 0, distance = 0, seek_count = 0, curr_head = 0;
        cout<<"\n The seek sequence:\n";
        while(count != n)
   {
                int pos, min = INT_MAX;
                for( int i=0; i<n; i++ )
                {
                        curr_head = req_arr[i];
                        distance = abs(curr_head - head);
                        if(min > distance)
                        {
                                min = distance;
                                pos = i;
                        }
                }
      cout<<req_arr[pos]<<" ";
      seek_count += min;
      req_arr[pos] = INT_MAX;
      count = count + 1;
   }
```

```
        cout<<"\n The seek count= "<<seek_count;
}
int main()
{
        int n, head;
        vector<int> req_arr;
    cout<<"\n Enter the number of locations: ";
        cin>>n;
        for(int i=0 ; i<n ;i++)
        {
                int temp;
                cout<<"\n Enter location "<<i+1<<": ";
                cin>>temp;
                req_arr.push_back(temp);
        }
        cout<<"\n Enter Disk Head Pointer: ";
        cin>>head;
        sstfalgo(req_arr, head, n);
        return 0;
}
```

OUTPUT:

```
Enter the number of locations: 5

Enter location 1: 176

Enter location 2: 79

Enter location 3: 60

Enter location 4: 34

Enter location 5: 11

Enter Disk Head Pointer: 50

The seek sequence:
60 34 79 11 176
The seek count= 220
```

RESULT:

Hence, Shortest Seek Time First (SSTF) Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

(III) SCAN (Elevator) ALGORITHM:

AIM:

To implement the SCAN (Elevator) Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start
2. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. 'head' is the position of disk head.
3. Let direction represents whether the head is moving towards left or right.
4. In the direction in which the head is moving, service all tracks one by one.
5. Calculate the absolute distance of the track from the head.
6. Increment the total seek count with this distance.
7. Currently serviced track position now becomes the new head position.
8. Go to step 3 until we reach one of the ends of the disk.
9. If we reach the end of the disk, reverse the direction and go to step 2 until all tracks in the request array have not been serviced.
10. Stop.

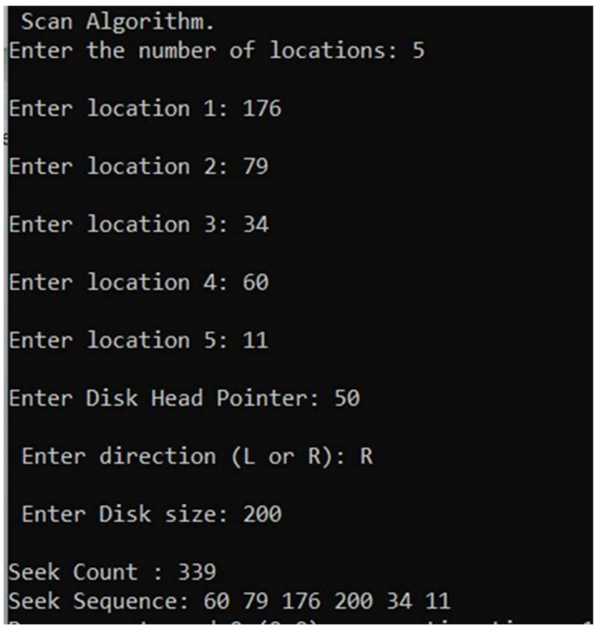
PROGRAM CODE:

```
#include<bits/stdc++.h>
using namespace std;
void scan(vector<int>, int, vector<int>, int , char, int);
int main()
{
        int n, head, seek_time, seek_count = 0, disk_size;
        vector<int> seek_sequence, reqarr;
        char dir;
        cout<<"\n Scan Algorithm.";
        cout<<"\nEnter the number of locations: ";
        cin>>n;
        for(int i=0 ; i<n ;i++)
```

```
        {
                int temp;
                cout<<"\nEnter location "<<i+1<<": ";
                cin>>temp;
                reqarr.push_back(temp);
        }
        cout<<"\nEnter Disk Head Pointer: ";
        cin>>head;
        cout<<"\n Enter direction (L or R): ";
        cin>>dir;
        cout<<"\n Enter Disk size: ";
        cin>>disk_size;
        scan(reqarr, head, seek_sequence, n, dir, disk_size);
        return 0;
}

void scan(vector<int> reqarr, int head, vector<int> seek_sequence, int n, char dir, int
disk_size)
{
        vector<int> left, right;
        for(int i=0 ; i<n ; i++)
        {
                if(reqarr[i] < head)
                        left.push_back(reqarr[i]);
                else if(reqarr[i] > head)
                        right.push_back(reqarr[i]);
        }
        int distance = 0, seek_count = 0, curr_pos = head;
        sort(right.begin(), right.end());
   sort(left.begin(), left.end(), greater<int>());
        if(dir == 'R')
   {
                for (int i = 0; i < right.size(); i++)
      {
         distance = abs(curr_pos - right[i]);
                    curr_pos = right[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
      }
      distance = abs(curr_pos - disk_size);
```

```
        curr_pos = disk_size;
        seek_sequence.push_back(curr_pos);
        seek_count = seek_count + distance;
            for(int i=0 ; i<left.size() ; i++)
            {
                    distance = abs(curr_pos - left[i]);
                    curr_pos = left[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
            }
    }
    else
        {
                for(int i=0 ; i<left.size() ; i++)
                {
                        distance = abs(curr_pos - left[i]);
                        curr_pos = left[i];
                        seek_sequence.push_back(curr_pos);
                        seek_count = seek_count + distance;
                }
                distance = abs(curr_pos - 0);
        curr_pos = 0;
        seek_sequence.push_back(curr_pos);
        seek_count = seek_count + distance;
                for(int i=0 ; i<right.size() ; i++)
                {
                        distance = abs(curr_pos - right[i]);
                        curr_pos = right[i];
                    seek_sequence.push_back(curr_pos);
                        seek_count = seek_count + distance;
            }
        }
        cout<<"\nSeek Count : "<<seek_count;
        cout<<"\nSeek Sequence: ";
        for(int i=0 ; i<n+1 ; i++)
                cout<<seek_sequence[i]<<" ";
}
```

OUTPUT:



RESULT:

Hence, SCAN (Elevator) Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

(IV) C - SCAN ALGORITHM:

AIM:

To implement the C - SCAN Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start
2. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. 'head' is the position of disk head.
3. The head services only in the right direction from 0 to the size of the disk.
4. While moving in the left direction do not service any of the tracks.
5. When we reach the beginning (left end) , reverse the direction.
6. While moving in the right direction it services all tracks one by one.
7. While moving in the right direction calculate the absolute distance of the track from the head.

8. Increment the total seek count with this distance.
9. Currently serviced track position now becomes the new head position.
10. Go to step 6 until we reach the right end of the disk.
11. If we reach the right end of the disk, reverse the direction and go to step 3 until all tracks in the request array have not been serviced.
12. Stop.

PROGRAM CODE:

```
#include<bits/stdc++.h>
using namespace std;
void clook(vector<int>, int, vector<int>, int , char, int);
int main()
{
	int n, head, seek_time, seek_count = 0, disk_size;
	vector<int> seek_sequence, reqarr;
	char dir;
	cout<<"\n C-SCAN Algorithm.";
	cout<<"\nEnter the number of locations: ";
	cin>>n;
	for(int i=0 ; i<n ;i++)
	{
		int temp;
		cout<<"\nEnter location "<<i+1<<": ";
		cin>>temp;
		reqarr.push_back(temp);
	}
	cout<<"\nEnter Disk Head Pointer: ";
	cin>>head;
	cout<<"\nEnter Disk Size: ";
	cin>>disk_size;
	cout<<"\n Enter direction (L or R): ";
	cin>>dir;
	clook(reqarr, head, seek_sequence, n, dir, disk_size);
	return 0;
}

void clook(vector<int> reqarr, int head, vector<int> seek_sequence, int n, char dir, int
disk_size)
```

```
{
        vector<int> left, right;
        for(int i=0 ; i<n ; i++)
        {
                if(reqarr[i] < head)
                        left.push_back(reqarr[i]);
                else if(reqarr[i] > head)
                        right.push_back(reqarr[i]);
        }
        int distance = 0, seek_count = 0, curr_pos = head;
        if(dir == 'R')
   {
      sort(left.begin(), left.end());
      sort(right.begin(), right.end());
                for (int i = 0; i < right.size(); i++)
      {
         distance = abs(curr_pos - right[i]);
                    curr_pos = right[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
      }
           distance = abs(curr_pos - 0);
           curr_pos = 0;
           seek_sequence.push_back(curr_pos);
           seek_count = seek_count + distance;
           for(int i=1 ; i<left.size() ; i++)
           {
                    distance = abs(curr_pos - left[i]);
                    curr_pos = left[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
           }
   }
   else
        {
           sort(left.begin(), left.end(), greater<int>());
      sort(right.begin(), right.end(), greater<int>());
                for(int i=0 ; i<left.size() ; i++)
                {
                        distance = abs(curr_pos - left[i]);
```

```
                curr_pos = left[i];
                seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
            }
            distance = abs(curr_pos - disk_size);
            curr_pos = disk_size;
            seek_sequence.push_back(curr_pos);
            seek_count = seek_count + distance;
            for(int i=1 ; i<right.size() ; i++)
            {
                distance = abs(curr_pos - right[i]);
                curr_pos = right[i];
              seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
          }
        }
        cout<<"\nSeek Count : "<<seek_count;
        cout<<"\nSeek Sequence: ";
        for(int i=0 ; i<n ; i++)
            cout<<seek_sequence[i]<<" ";}
```

OUTPUT:

```
 C-SCAN Algorithm.
Enter the number of locations: 5

Enter location 1: 176

Enter location 2: 79

Enter location 3: 60

Enter location 4: 34

Enter location 5: 11

Enter Disk Head Pointer: 50

Enter Disk Size: 200

 Enter direction (L or R): R

Seek Count : 336
Seek Sequence: 60 79 176 0 34
```

RESULT:

Hence, C - SCAN Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

(V) LOOK ALGORITHM:

AIM:

To implement the LOOK Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start
2. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. 'head' is the position of disk head.
3. The initial direction in which the head is moving is given and it services in the same direction.
4. The head services all the requests one by one in the direction the head is moving.
5. The head continues to move in the same direction until all the requests in this direction are finished.
6. While moving in this direction calculate the absolute distance of the track from the head.
7. Increment the total seek count with this distance.
8. Currently serviced track position now becomes the new head position.
9. Go to step 5 until we reach the last request in this direction.
10. If we reach where no requests are needed to be serviced in this direction, reverse the direction and go to step 3 until all tracks in the request array have not been serviced.
11. Stop

PROGRAM CODE:

```
#include<bits/stdc++.h>
using namespace std;
void look(vector<int>, int, vector<int>, int , char);
int main()
{
```

```
        int n, head, seek_time, seek_count = 0;
        vector<int> seek_sequence, reqarr;
        char dir;
        cout<<"\n LOOK Algorithm.";
        cout<<"\nEnter the number of locations: ";
        cin>>n;
        for(int i=0 ; i<n ;i++)
        {
                int temp;
                cout<<"\nEnter location "<<i+1<<": ";
                cin>>temp;
                reqarr.push_back(temp);
        }
        cout<<"\nEnter Disk Head Pointer: ";
        cin>>head;
        cout<<"\n Enter direction (L or R): ";
        cin>>dir;
        look(reqarr, head, seek_sequence, n, dir);
        return 0;
}
void look(vector<int> reqarr, int head, vector<int> seek_sequence, int n, char dir)
{
        vector<int> left, right;
        for(int i=0 ; i<n ; i++)
        {
                if(reqarr[i] < head)
                        left.push_back(reqarr[i]);
                else if(reqarr[i] > head)
                        right.push_back(reqarr[i]);
        }
        int distance = 0, seek_count = 0, curr_pos = head;
        if(dir == 'R')
   {
                sort(left.begin(), left.end(), greater<int>());
                sort(right.begin(), right.end());
                for (int i = 0; i < right.size(); i++)
      {
         distance = abs(curr_pos - right[i]);
                    curr_pos = right[i];
                    seek_sequence.push_back(curr_pos);
```

```
                seek_count = seek_count + distance;
        }

            for(int i=0 ; i<left.size() ; i++)
            {
                distance = abs(curr_pos - left[i]);
                curr_pos = left[i];
                seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
            }
    }
    else
        {
            sort(right.begin(), right.end(), greater<int>());
            sort(left.begin(), left.end());
            for(int i=0 ; i<left.size() ; i++)
            {
                distance = abs(curr_pos - left[i]);
                curr_pos = left[i];
                seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
            }
            for(int i=0 ; i<right.size() ; i++)
            {
                distance = abs(curr_pos - right[i]);
                curr_pos = right[i];
              seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
            }
        }
        cout<<"\nSeek Count : "<<seek_count;
        cout<<"\nSeek Sequence: ";
        for(int i=0 ; i<n ; i++)
            cout<<seek_sequence[i]<<" ";
}
```

OUTPUT:

```
 LOOK Algorithm.
Enter the number of locations: 5

Enter location 1: 176

Enter location 2: 79

Enter location 3: 60

Enter location 4: 34

Enter location 5: 90

Enter Disk Head Pointer: 50

 Enter direction (L or R): R

Seek Count : 268
Seek Sequence: 60 79 90 176 34
```

RESULT:

Hence, LOOK Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

(VI) C - LOOK ALGORITHM:

AIM:

To implement the C - LOOK Disk Scheduling Algorithm in C.

ALGORITHM:

1. Start
2. Let Request array represents an array storing indexes of the tracks that have been requested in ascending order of their time of arrival and head is the position of the disk head.
3. The initial direction in which the head is moving is given and it services in the same direction.
4. The head services all the requests one by one in the direction it is moving.
5. The head continues to move in the same direction until all the requests in this direction have been serviced.

6. While moving in this direction, calculate the absolute distance of the tracks from the head.
7. Increment the total seek count with this distance.
8. Currently serviced track position now becomes the new head position.
9. Go to step 5 until we reach the last request in this direction.
10. If we reach the last request in the current direction then reverse the direction and move the head in this direction until we reach the last request that is needed to be serviced in this direction without servicing the intermediate requests.
11. Reverse the direction and go to step 3 until all the requests have not been serviced.
12. Stop

PROGRAM CODE:

```
#include<bits/stdc++.h>
using namespace std;
void clook(vector<int>, int, vector<int>, int , char);
int main()
{
	int n, head, seek_time, seek_count = 0;
	vector<int> seek_sequence, reqarr;
	char dir;
	cout<<"\n C-LOOK Algorithm.";
	cout<<"\nEnter the number of locations: ";
	cin>>n;
	for(int i=0 ; i<n ;i++)
	{
		int temp;
		cout<<"\nEnter location "<<i+1<<": ";
		cin>>temp;
		reqarr.push_back(temp);
	}
	cout<<"\nEnter Disk Head Pointer: ";
	cin>>head;
	cout<<"\n Enter direction (L or R): ";
	cin>>dir;
	clook(reqarr, head, seek_sequence, n, dir);
	return 0;
}
```

```
void clook(vector<int> reqarr, int head, vector<int> seek_sequence, int n, char dir)
{
        vector<int> left, right;
        for(int i=0 ; i<n ; i++)
        {
                if(reqarr[i] < head)
                        left.push_back(reqarr[i]);
                else if(reqarr[i] > head)
                        right.push_back(reqarr[i]);
        }
        sort(left.begin(), left.end());
        sort(right.begin(), right.end());
        int distance = 0, seek_count = 0, curr_pos = head;
        if(dir == 'R')
   {
                for (int i = 0; i < right.size(); i++)
      {
         distance = abs(curr_pos - right[i]);
                    curr_pos = right[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
      }
           distance = abs(curr_pos - left[0]);
           curr_pos = left[0];
           seek_sequence.push_back(curr_pos);
           seek_count = seek_count + distance;
           for(int i=1 ; i<left.size() ; i++)
           {
                    distance = abs(curr_pos - left[i]);
                    curr_pos = left[i];
                    seek_sequence.push_back(curr_pos);
                    seek_count = seek_count + distance;
           }
   }
   else
        {
                for(int i=0 ; i<left.size() ; i++)
                {
                        distance = abs(curr_pos - left[i]);
```

```
                curr_pos = left[i];
                seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
            }
            distance = abs(curr_pos - right[0]);
            curr_pos = right[0];
            seek_sequence.push_back(curr_pos);
            seek_count = seek_count + distance;
            for(int i=1 ; i<right.size() ; i++)
            {
                distance = abs(curr_pos - right[i]);
                curr_pos = right[i];
              seek_sequence.push_back(curr_pos);
                seek_count = seek_count + distance;
          }
        }
        cout<<"\nSeek Count : "<<seek_count;
        cout<<"\nSeek Sequence: ";
        for(int i=0 ; i<n ; i++)
            cout<<seek_sequence[i]<<" ";
}
```

OUTPUT:

```
 C-LOOK Algorithm.
Enter the number of locations: 5

Enter location 1: 176

Enter location 2: 79

Enter location 3: 34

Enter location 4: 60

Enter location 5: 11

Enter Disk Head Pointer: 50

 Enter direction (L or R): R

Seek Count : 314
Seek Sequence: 60 79 176 11 34
```

RESULT:

Hence, C - LOOK Disk Scheduling Algorithm has been successfully executed and Output verified successfully.

EXP NO: 9 DATE : 03-06-2022	FAMILIARIZATION WITH PAGE REPLACEMENT ALGORITHM

(I) Optimal Page Replacement ALGORITHM:

AIM:

To implement the Optimal Page replacement Algorithm in C.

ALGORITHM:

1. Start the process.
2. Get user input of the number of pages.
3. Get user input of Page number sequence as an array.
4. Get user input of Number of frames.
5. Initialise all frames to -1.
6. Traverse the array of page number sequence.
7. Check for page fault element by element.
8. In case of page fault, replace the page by the most optimal page in the frame array by comparing in the reference string .
9. Display the values of frames and total page fault.

PROGRAM CODE:

```
#include<stdio.h>
int main()
{
        int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2,
flag3, i;
         int j,k, pos, max, faults = 0;
        printf("Enter number of frames: ");
        scanf("%d", &no_of_frames);

        printf("Enter number of pages: ");
        scanf("%d", &no_of_pages);

        printf("Enter page reference string: ");

        for(i = 0; i < no_of_pages; ++i){
```

```
scanf("%d", &pages[i]);
}
for(i = 0; i < no_of_frames; ++i){
frames[i] = -1;
}
for(i = 0; i < no_of_pages; ++i){
flag1 = flag2 = 0;
for(j = 0; j < no_of_frames; ++j){
if(frames[j] == pages[i]){
        flag1 = flag2 = 1;
        break;
        }
}

if(flag1 == 0){
for(j = 0; j < no_of_frames; ++j){
        if(frames[j] == -1){
        faults++;
        frames[j] = pages[i];
        flag2 = 1;
        break;
        }
}
}
if(flag2 == 0){
flag3 =0;

for(j = 0; j < no_of_frames; ++j){
temp[j] = -1;

for(k = i + 1; k < no_of_pages; ++k){
if(frames[j] == pages[k]){
temp[j] = k;
break;
}
}
}

for(j = 0; j < no_of_frames; ++j){
if(temp[j] == -1){
```

```
            pos = j;
            flag3 = 1;
            break;
            }
            }

            if(flag3 ==0){
            max = temp[0];
            pos = 0;

            for(j = 1; j < no_of_frames; ++j){
            if(temp[j] > max){
            max = temp[j];
            pos = j;
            }
            }
            }
    frames[pos] = pages[i];
    faults++;
            }

            printf("\n");

            for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
            }
            }

            printf("\n\nTotal Page Faults = %d", faults);

            return 0;
    }
```

OUTPUT:

```
rusa@rusa-Veriton-M4660G:~$ gedit ora.c
rusa@rusa-Veriton-M4660G:~$ cc ora.c
rusa@rusa-Veriton-M4660G:~$ ./a.out
Enter number of frames: 3
Enter number of pages: 12
Enter page reference string: 0 1 2 3 0 1 4 0 1 2 3 4

0       -1      -1
0       1       -1
0       1       2
0       1       3
0       1       3
0       1       3
0       1       4
0       1       4
0       1       4
2       1       4
3       1       4
3       1       4

Total Page Faults = 7
```

RESULT:

Hence, Optimal page replacement Algorithm has been successfully executed and output verified successfully.

(II) Least Recently Used (LRU):

AIM:

To implement the Least Recently Used (LRU) page replacement Algorithm in C.

ALGORITHM :

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least recently used page by counter value
7. Stack them according to the selection.
8. Display the values
9. Stop the process

PROGRAM CODE:

```
#include<stdio.h>
main()
{
int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
printf("Enter no of pages:");
scanf("%d",&n);
printf("Enter the reference string:");
for(i=0;i<n;i++)
        scanf("%d",&p[i]);
printf("Enter no of frames:");
scanf("%d",&f);
q[k]=p[k];
printf("\n\t%d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++)
        {
                c1=0;
                for(j=0;j<f;j++)
                {
                        if(p[i]!=q[j])
                        c1++;
                }
                if(c1==f)
                {
                        c++;
                        if(k<f)
                        {
                                q[k]=p[i];
                                k++;
                                for(j=0;j<k;j++)
                                printf("\t%d",q[j]);
                                printf("\n");
                        }
                        else
                        {
                                for(r=0;r<f;r++)
                                {
                                        c2[r]=0;
```

```
                                        for(j=i-1;j<n;j--)
                                        {
                                        if(q[r]!=p[j])
                                        c2[r]++;
                                        else
                                        break;
                                }
                        }
                        for(r=0;r<f;r++)
                         b[r]=c2[r];
                        for(r=0;r<f;r++)
                        {
                                for(j=r;j<f;j++)
                                {
                                        if(b[r]<b[j])
                                        {
                                                t=b[r];
                                                b[r]=b[j];
                                                b[j]=t;
                                        }
                                }
                        }
                        for(r=0;r<f;r++)
                        {
                                if(c2[r]==b[0])
                                q[r]=p[i];
                                printf("\t%d",q[r]);
                        }
                        printf("\n");
                }
        }
}
printf("\nThe no of page faults is %d",c);
}
```

OUTPUT:

```
rusa@rusa-Veriton-M4660G:~$ cc lru.c
rusa@rusa-Veriton-M4660G:~$ ./a.out
Enter number of frames: 3
Enter number of pages: 12
Enter reference string:  0 1 2 3 0 1 4 0 1 2 3 4

0       -1      -1
0       1       -1
0       1       2
3       1       2
3       0       2
3       0       1
4       0       1
4       0       1
4       0       1
2       0       1
2       3       1
2       3       4

Total Page Faults = 10rusa@rusa-Veriton-M4660G:~$ gedit ora.c
```

RESULT:

Hence, LRU page replacement Algorithm has been successfully run and output verified successfully.

(III) First in First Out (FIFO):

AIM:

To implement First in First Out (FIFO) page replacement Algorithm in C.

ALGORITHM:

1. Start the process
2. Declare the size with respect to page length
3. Check the need of replacement from the page to memory
4. Check the need of replacement from old page to new page in memory
5. Form a queue to hold all pages
6. Insert the page require memory into the queue
7. Check for bad replacement and page fault
8. Get the number of processes to be inserted
9. Display the values
10. Stop the process

PROGRAM CODE:

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n Enter the number of pages : ");
scanf("%d",&n);
printf("\n Enter the page number :  ");
for(i=1;i<=n;i++)
      scanf("%d",&a[i]);
      printf("\nEnter the number of frames  : ");
      scanf("%d",&no);
for(i=0;i<no;i++)
      frame[i]= -1;
      j=0;
      printf("\tRef String\t Page Frames\n");
for(i=1;i<=n;i++)
{
      printf("%d\t\t",a[i]);
      avail=0;
         for(k=0;k<no;k++)
         if(frame[k]==a[i])
               avail=1;
           if (avail==0)
           {
                 frame[j]=a[i];
                 j=(j+1)%no;
                  count++;
                  for(k=0;k<no;k++)
                           printf("%d\t",frame[k]);
         }
         printf("\n");
}
printf("Page Fault Is %d",count);
return 0;
}
```

OUTPUT:

```
Enter the total number of pages: 12
Enter the page number sequence: 1 2 3 4 1 2 5 1 2 3 4 5
Enter the number of frames: 3

Reference string                Page frames
        1                       1       -1      -1
        2                       1       2       -1
        3                       1       2       3
        4                       4       2       3
        1                       4       1       3
        2                       4       1       2
        5                       5       1       2
        1
        2
        3                       5       3       2
        4                       5       3       4
        5

Page Fault Is 9
```

RESULT:

Hence, FIFO page replacement Algorithm has been successfully run and output verified successfully.