LINUX BASIC COMMANDS

cd → Change to new directory
mkdir → create new directory
rmdir → remove empty directory (remove files first)
mv → change name of directory
pwd → show current directory
date → show date and time
history → list of previously executed commands
cal month year → Prints a calendar for the specified month of the specified year.
man → show online documentation by program name
w, who → who is on the system and what they are doing
who am i → who is logged onto this terminal
uptime → show one line summary of system status
finger → find out info about a user@system
whois → look up information in the Stanford Directory
tty → know the terminal name.
uname → print system information
cat → view files
cp →copy files
ls → list files in a directory and their attributes
mv → change file name or directory location
rm → remove files
head →show first few lines of a file(s)
tail →show last few lines of a file; or reverse line order
vi → full-featured screen editor for modifying text files
echo $$ → process id of current shell.
ps → process status
kill → kill background job or previous process…

Exercise 2 – Some Shell Scripts

16BCE0783 – Daksh

(Screenshots Together at Last)

1. Hello World, Date, User and Current Directory

```
#!/bin/bash

echo "Hello $LOGNAME!"
echo "Current Date: $(date)"
echo "User is: $(whoami)"
echo "Current Directory is: $PWD"
```

2. Largest of Three Numbers

```
echo "Enter three numbers with spaces!"
read a b c
l=$a
if [ $l -gt $b ]
then
l=$b
fi
if [ $c -gt $l ]
then
l=$c
fi
echo "Largest of $a $b and $c is $l"
```

3. Two Numbers Swapping

```
#!/bin/bash

echo "Enter First Number"
read a
echo "Enter Second Number"
read b
echo "a and b before swapping are $a and $b"
a=$((a+b))
b=$((a-b))
```

```
a=$((a-b))
echo "a and b after swapping are $a and $b"
```

4. Finding Average Marks and Grade of a Student

```
#!/bin/bash

echo "Enter the Name of the Student"
read name
echo "Enter the Registration Number"
read reg
echo "Enter the marks separated by the space"
read m1 m2 m3 m4 m5
echo "Name of the Student is $name"
echo "Registration Number of the $reg"
echo "Marks Obtained"
echo "M1 M2 M3 M4 M5"
echo $m1 $m2 $m3 $m4 $m5
per=`echo \($m1+$m2+$m3+$m4+$m5\) /5|bc`
echo "Average is $per"
if test $per -ge 60
then
echo "Grade: First"
elif test $per -ge 50 -a $per -le 59
then
echo "Grade: Second"
elif test $per -ge 40 -a $per -le 49
then
echo "Grade: Third"
else
echo "Fail!!!"
fi
```

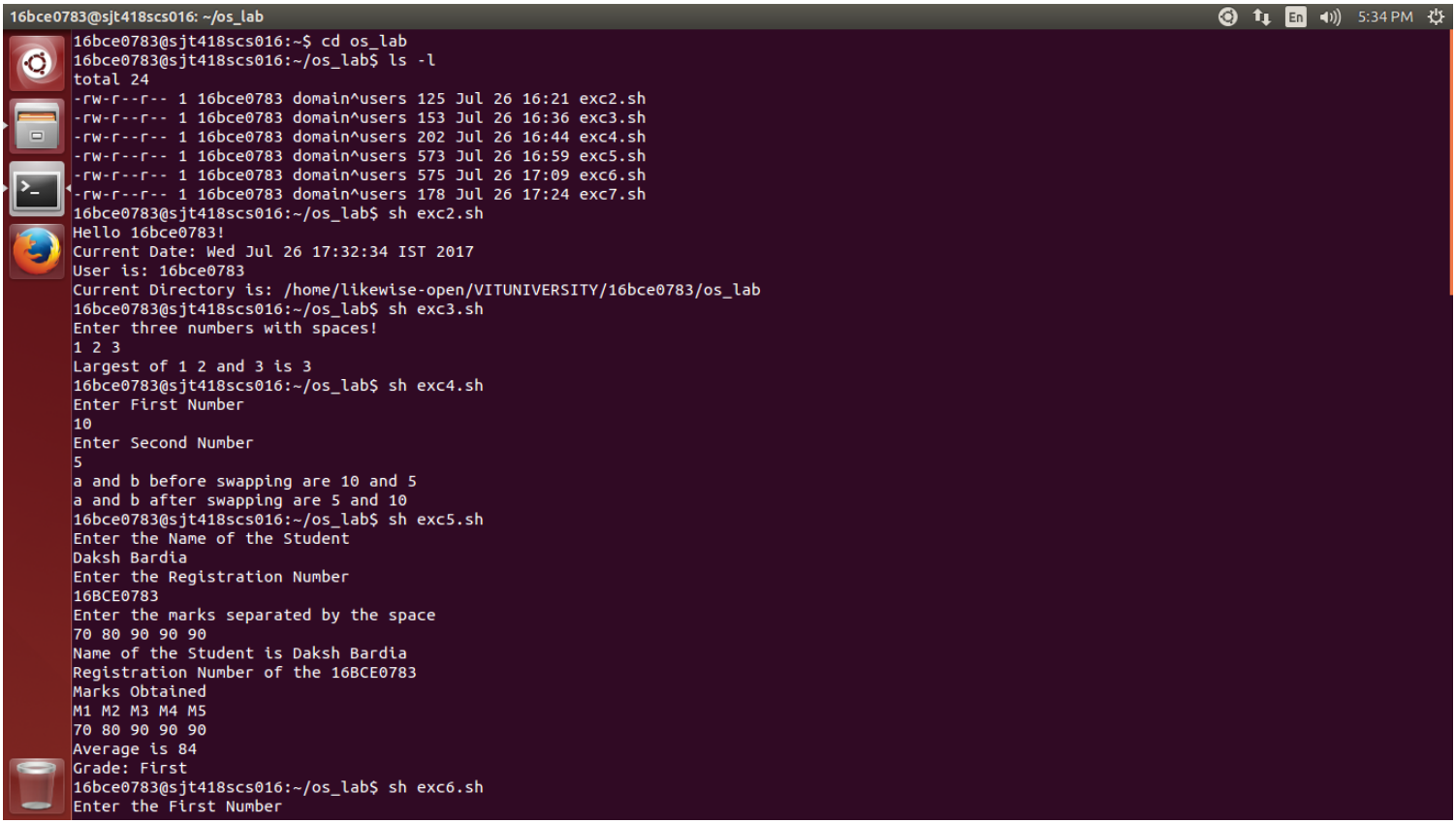
5. Menu Based Addition, Substration, Multiplication and Division

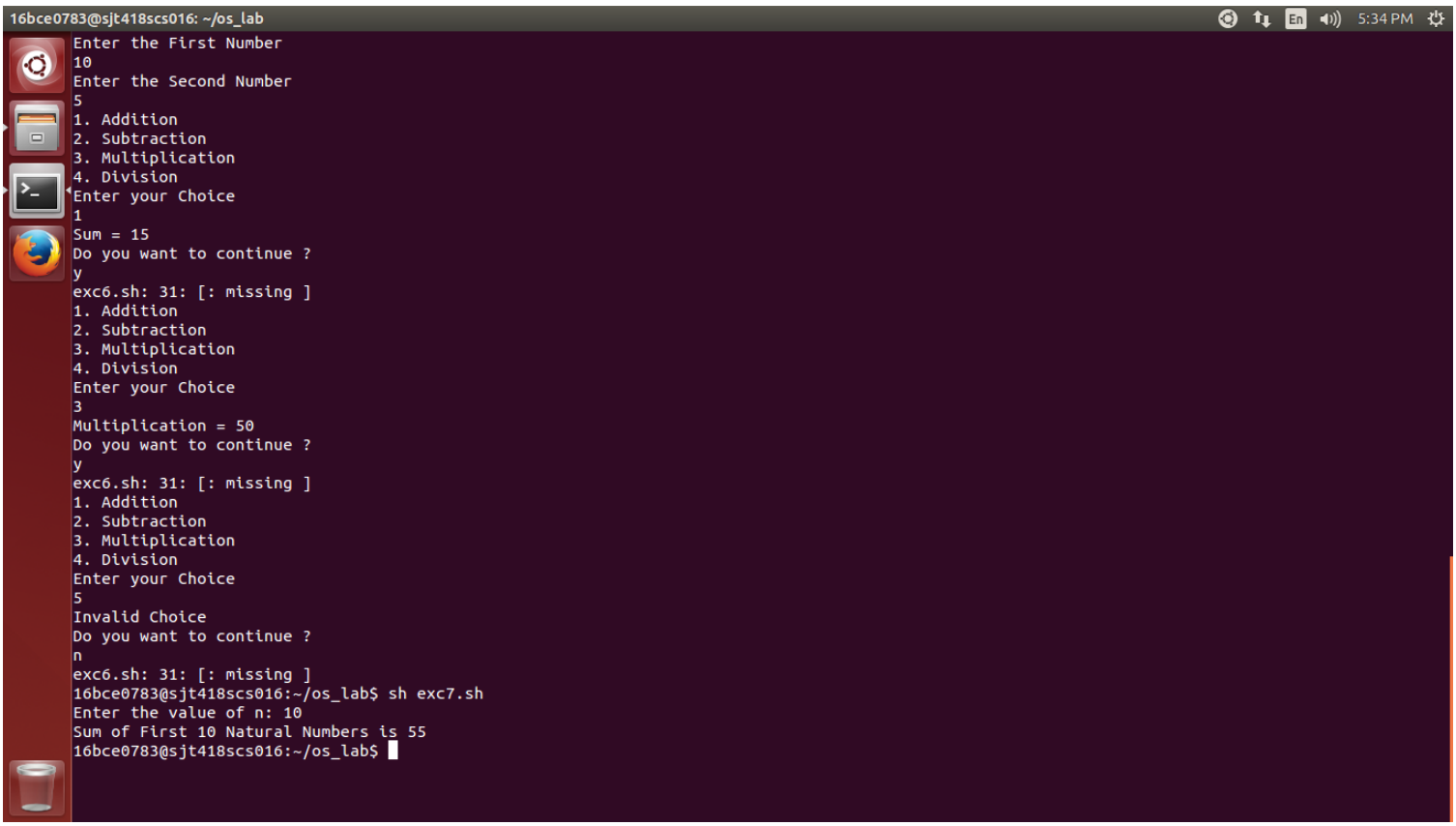
```
#!/bin/bash

clear
sum=0
i="y"
echo "Enter the First Number"
read n1
echo "Enter the Second Number"
```

```
read n2
while [ $i = "y" ]
do
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
echo "Enter your Choice"
read ch
case $ch in
    1) sol=`expr $n1 + $n2`
    echo "Sum = $sol";;
    2) sol=`expr $n1 - $n2`
    echo "Subtraction = $sol";;
    3) sol=`expr $n1 \* $n2`
    echo "Multiplication = $sol";;
    4) sol=`expr $n1 / $n2`
    echo "Division = $sol";;
    *)echo "Invalid Choice";;
esac
echo "Do you want to continue ?"
read i
if [ $i = "y"]
then
    exit
fi
done
```

6. Sum of First n Natural Numbers

```
#!/bin/bash

echo -n "Enter the value of n: "
read n
i=1
sum=0
while [ $i -le $n ]
do
    sum=$(( $sum + $i ))
    i=$(( $i + 1 ))
done

echo "Sum of First $n Natural Numbers is $sum"
```


16bce0783@sjt418scs016: ~/os_lab
5:34 PM
16bce0783@sjt418scs016:~$ cd os_lab
16bce0783@sjt418scs016:~/os_lab$ ls -l
total 24
-rw-r--r-- 1 16bce0783 domain^users 125 Jul 26 16:21 exc2.sh
-rw-r--r-- 1 16bce0783 domain^users 153 Jul 26 16:36 exc3.sh
-rw-r--r-- 1 16bce0783 domain^users 202 Jul 26 16:44 exc4.sh
-rw-r--r-- 1 16bce0783 domain^users 573 Jul 26 16:59 exc5.sh
-rw-r--r-- 1 16bce0783 domain^users 575 Jul 26 17:09 exc6.sh
-rw-r--r-- 1 16bce0783 domain^users 178 Jul 26 17:24 exc7.sh
16bce0783@sjt418scs016:~/os_lab$ sh exc2.sh
Hello 16bce0783!
Current Date: Wed Jul 26 17:32:34 IST 2017
User is: 16bce0783
Current Directory is: /home/likewise-open/VITUNIVERSITY/16bce0783/os_lab
16bce0783@sjt418scs016:~/os_lab$ sh exc3.sh
Enter three numbers with spaces!
1 2 3
Largest of 1 2 and 3 is 3
16bce0783@sjt418scs016:~/os_lab$ sh exc4.sh
Enter First Number
10
Enter Second Number
5
a and b before swapping are 10 and 5
a and b after swapping are 5 and 10
16bce0783@sjt418scs016:~/os_lab$ sh exc5.sh
Enter the Name of the Student
Daksh Bardia
Enter the Registration Number
16BCE0783
Enter the marks separated by the space
70 80 90 90 90
Name of the Student is Daksh Bardia
Registration Number of the 16BCE0783
Marks Obtained
M1 M2 M3 M4 M5
70 80 90 90 90
Average is 84
Grade: First
16bce0783@sjt418scs016:~/os_lab$ sh exc6.sh
Enter the First Number

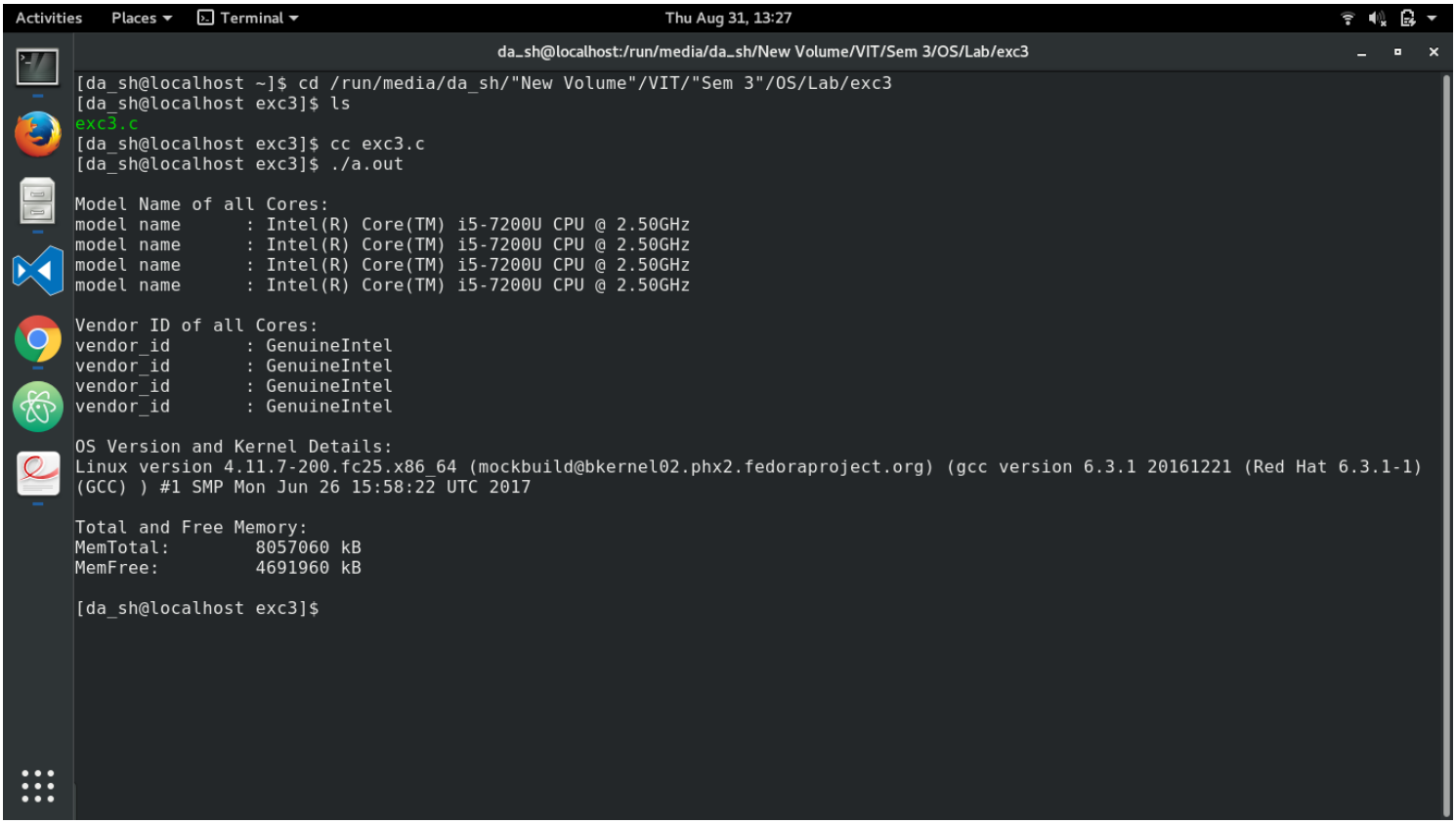

16bce0783@sjt418scs016: ~/os_lab
5:34 PM
Enter the First Number
10
Enter the Second Number
5
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your Choice
1
Sum = 15
Do you want to continue ?
y
exc6.sh: 31: [: missing]
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your Choice
3
Multiplication = 50
Do you want to continue ?
y
exc6.sh: 31: [: missing]
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your Choice
5
Invalid Choice
Do you want to continue ?
n
exc6.sh: 31: [: missing]
16bce0783@sjt418scs016:~/os_lab$ sh exc7.sh
Enter the value of n: 10
Sum of First 10 Natural Numbers is 55
16bce0783@sjt418scs016:~/os_lab$

Exercise 3 – System Info thru Linux Commands

16BCE0783 – Daksh

Displaying Some System Information using Linux Commands on Linux Fedora:

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

void main(){
    printf("\nModel Name of all Cores:\n");
    system("cat /proc/cpuinfo | grep 'model name'");
    printf("\nVendor ID of all Cores: \n");
    system("cat /proc/cpuinfo | grep vendor_id");
    printf("\nOS Version and Kernel Details: \n");
    system("cat /proc/version");
    printf("\nTotal and Free Memory: \n");
    system("cat /proc/meminfo | grep MemTotal");
    system("cat /proc/meminfo | grep MemFree");
    printf("\n");
}
```

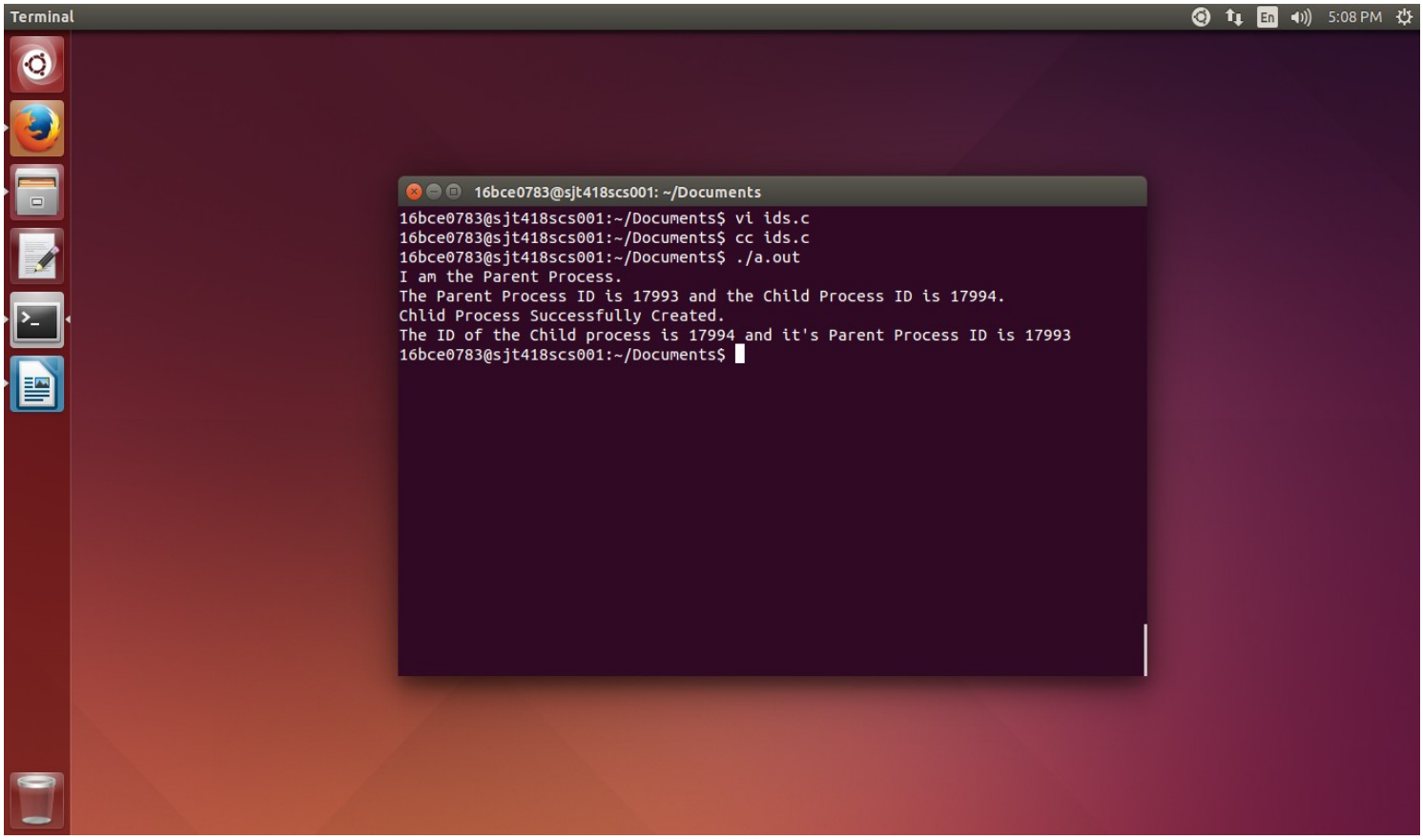


Exercise 4 – Process Creation

16BCE0783 – Daksh

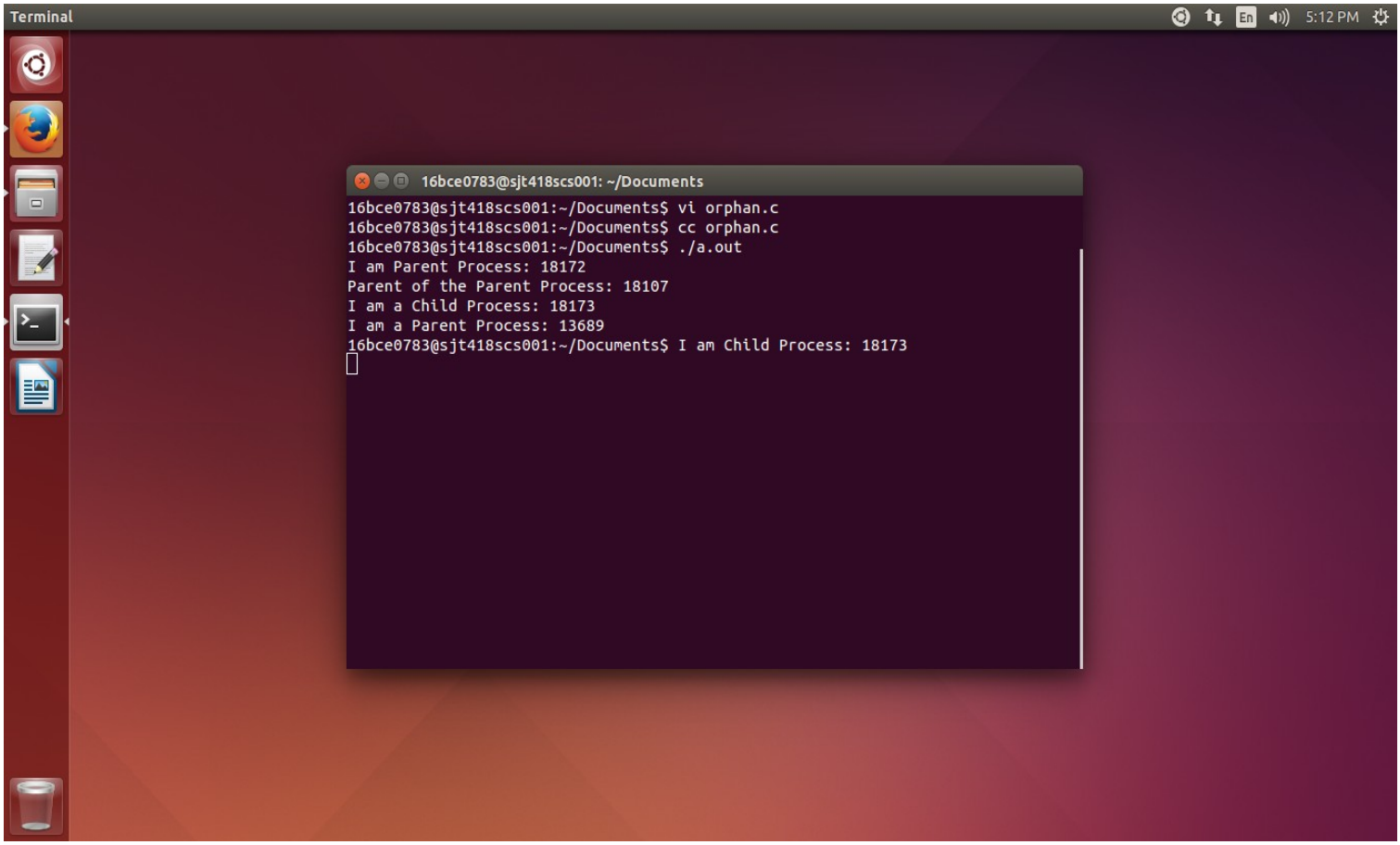
1. Parent Process Creation

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

void main(){
	int id = fork();
	if (id < 0){
		printf("Child Process Creation Failed !!\n");
	}
	else if (id == 0){
		printf("Chlid Process Successfully Created.\nThe ID of the Child process is %d and
it's Parent Process ID is %d\n",getpid(),getppid());
	}
	else{
		printf("I am the Parent Process.\n");
		printf("The Parent Process ID is %d and the Child Process ID is %d.\n",getpid(),id);
	}
}
```



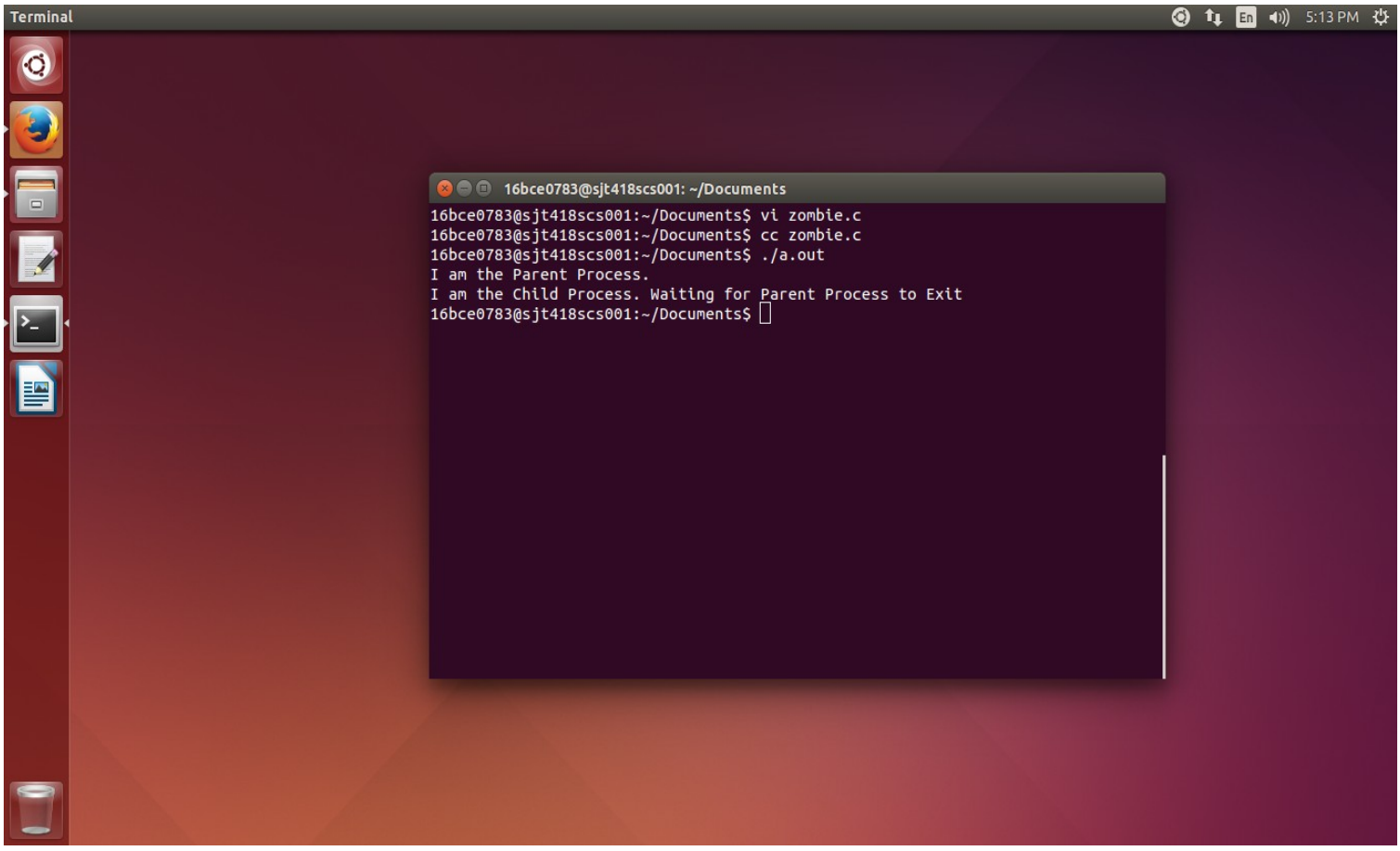
2. Orphan Process Creation

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

void main(){
        int id = fork();
        if (id<0){
                printf("Process Creation Failed! Error!");
                exit(0);
        }
        else if(id==0){
                printf("I am a Child Process: %d\n",getpid());
                printf("I am a Parent Process: %d\n",getppid());
                sleep(5);
                printf("I am Child Process: %d\n",getpid());
        }
        else{
                printf("I am Parent Process: %d\n",getpid());
                printf("Parent of the Parent Process: %d\n",getppid());
        }
}
```



3. Zombie Process Creation

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void main(){
        int id = fork();
        if (id == 0){
                printf("I am the Child Process. Waiting for Parent Process to Exit\n");
                exit(0);
        }
        else if (id > 0) {
                printf("I am the Parent Process.\n");
                sleep(10);
        }
        else{
                printf("Child Process Creation Failed !\n");
        }
}
```



Exercise 5 – Non Preemptive Scheduling Algorithms

16BCE0783 – Daksh

Output Screenshots at Last

1. First Come First Serve (FCFS) Scheduling Algorithm

```c
#include<stdio.h>

void main(){
    int n,burst_time[20],waiting_time[20],total_at[20],avg_waiting_time=0,avg_total_at=0,i,j;
    printf("Enter the total number of processes: ");
    scanf("%d",&n);
    printf("\nEnter the Process Burst Time one by one: \n");
    for(i=0;i<n;i++){
        printf("P[%d]:",i+1);
        scanf("%d",&burst_time[i]);
    }
    waiting_time[0]=0;
    for(i=1;i<n;i++){
        waiting_time[i]=0;
        for(j=0;j<i;j++)
            waiting_time[i]+=burst_time[j];
    }
    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++){
        total_at[i]=burst_time[i]+waiting_time[i];
        avg_waiting_time+=waiting_time[i];
        avg_total_at+=total_at[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,burst_time[i],waiting_time[i],total_at[i]);
    }
    avg_waiting_time/=i;
    avg_total_at/=i;
    printf("\n\nAverage Waiting Time is: %d",avg_waiting_time);
    printf("\nAverage Turnaround Time is: %d\n\n",avg_total_at);
}
```

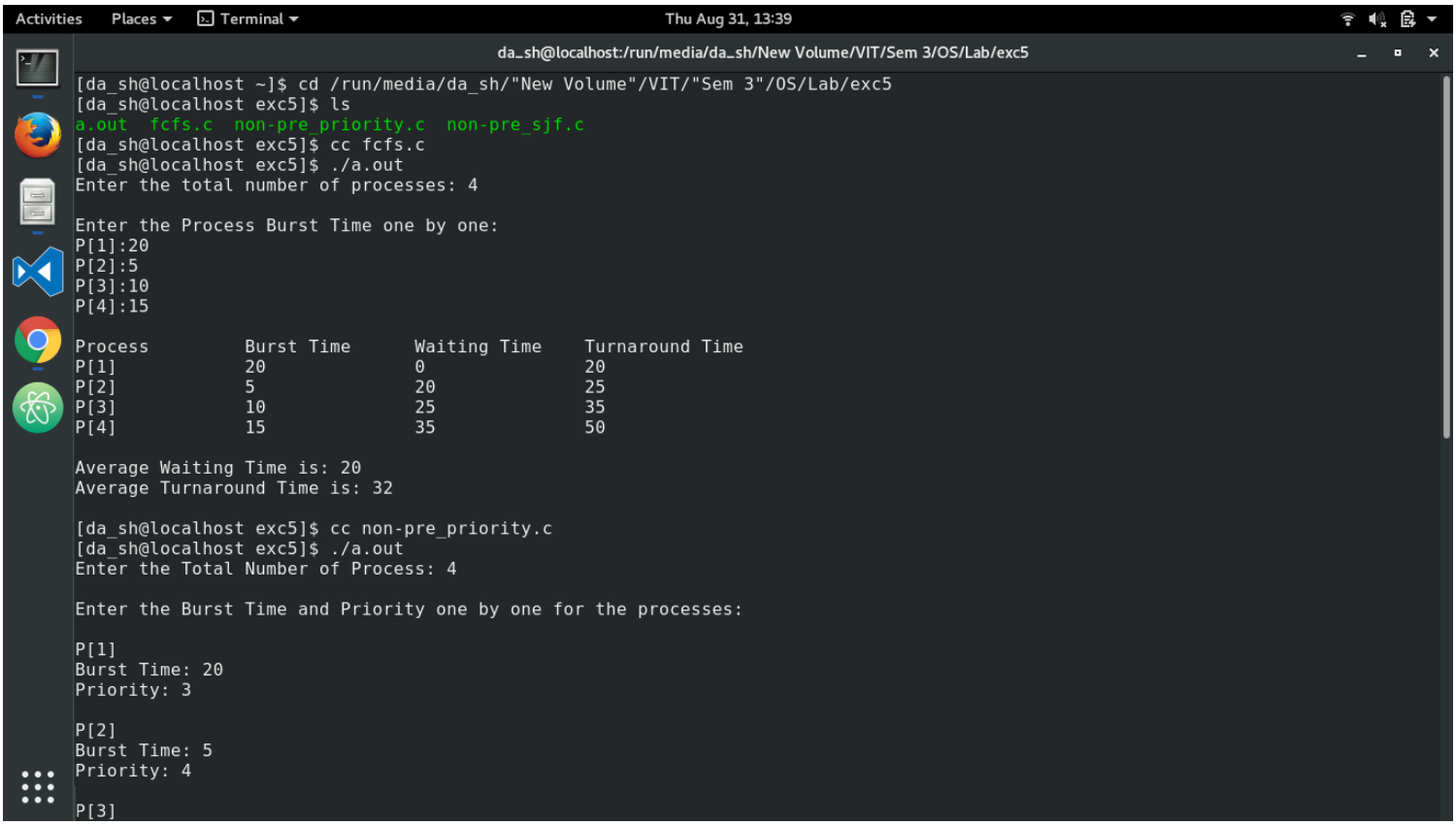
2. Non Preemptive Priority Scheduling Algorithm

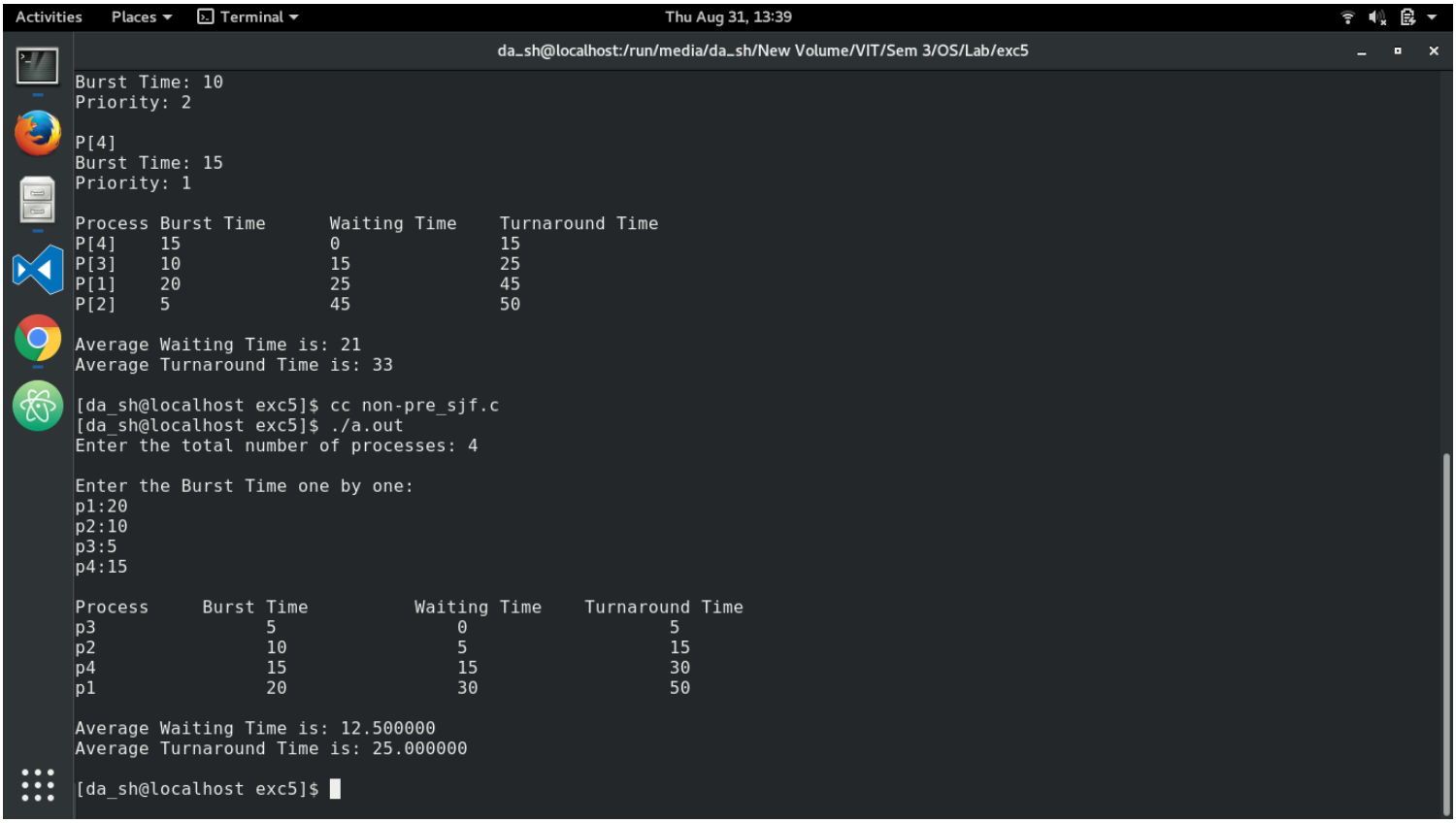
```c
#include<stdio.h>

void main(){
```

```
    int
burst_time[20],p[20],waiting_time[20],total_at[20],priority[20],i,j,n,total=0,pos,temp,avg_wa
iting_time,avg_total_at;
    printf("Enter the Total Number of Process: ");
    scanf("%d",&n);
    printf("\nEnter the Burst Time and Priority one by one for the processes: \n");
    for(i=0;i<n;i++){
        printf("\nP[%d]\n",i+1);
        printf("Burst Time: ");
        scanf("%d",&burst_time[i]);
        printf("Priority: ");
        scanf("%d",&priority[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++){
        pos=i;
        for(j=i+1;j<n;j++){
            if(priority[j]<priority[pos])
                pos=j;
        }
        temp=priority[i];
        priority[i]=priority[pos];
        priority[pos]=temp;
        temp=burst_time[i];
        burst_time[i]=burst_time[pos];
        burst_time[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }
    waiting_time[0]=0;
    for(i=1;i<n;i++){
        waiting_time[i]=0;
        for(j=0;j<i;j++)
            waiting_time[i]+=burst_time[j];
        total+=waiting_time[i];
    }
    avg_waiting_time=total/n;
    total=0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++){
        total_at[i]=burst_time[i]+waiting_time[i];
        total+=total_at[i];
        printf("\nP[%d]\t%d\t\t%d\t\t%d",p[i],burst_time[i],waiting_time[i],total_at[i]);
    }
    avg_total_at=total/n;
    printf("\n\nAverage Waiting Time is: %d",avg_waiting_time);
    printf("\nAverage Turnaround Time is: %d\n\n",avg_total_at);
}
```

3. Shortest Job First (SJF) Scheduling Algorithm

```c
#include<stdio.h>

void main(){
    int burst_time[20],p[20],waiting_time[20],total_at[20],i,j,n,total=0,pos,temp;
    float avg_waiting_time,avg_total_at;
    printf("Enter the total number of processes: ");
    scanf("%d",&n);
    printf("\nEnter the Burst Time one by one: \n");
    for(i=0;i<n;i++){
        printf("p%d:",i+1);
        scanf("%d",&burst_time[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++){
        pos=i;
        for(j=i+1;j<n;j++){
            if(burst_time[j]<burst_time[pos])
                pos=j;
        }
        temp=burst_time[i];
        burst_time[i]=burst_time[pos];
        burst_time[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }
    waiting_time[0]=0;
    for(i=1;i<n;i++){
        waiting_time[i]=0;
        for(j=0;j<i;j++)
            waiting_time[i]+=burst_time[j];
        total+=waiting_time[i];
    }
    avg_waiting_time=(float)total/n;
    total=0;
    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++){
        total_at[i]=burst_time[i]+waiting_time[i];
        total+=total_at[i];
        printf("\np%d\t\t  %d\t\t
%d\t\t\t%d",p[i],burst_time[i],waiting_time[i],total_at[i]);
    }
    avg_total_at=(float)total/n;
    printf("\n\nAverage Waiting Time is: %f",avg_waiting_time);
    printf("\nAverage Turnaround Time is: %f\n\n",avg_total_at);
}
```


Activities
Places
Terminal
Thu Aug 31, 13:39
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc5
[da_sh@localhost ~]$ cd /run/media/da_sh/"New Volume"/VIT/"Sem 3"/OS/Lab/exc5
[da_sh@localhost exc5]$ ls
a.out fcfs.c non-pre_priority.c non-pre_sjf.c
[da_sh@localhost exc5]$ cc fcfs.c
[da_sh@localhost exc5]$ ./a.out
Enter the total number of processes: 4
Enter the Process Burst Time one by one:
P[1]:20
P[2]:5
P[3]:10
P[4]:15
Process Burst Time Waiting Time Turnaround Time
P[1] 20 0 20
P[2] 5 20 25
P[3] 10 25 35
P[4] 15 35 50
Average Waiting Time is: 20
Average Turnaround Time is: 32
[da_sh@localhost exc5]$ cc non-pre_priority.c
[da_sh@localhost exc5]$ ./a.out
Enter the Total Number of Process: 4
Enter the Burst Time and Priority one by one for the processes:
P[1]
Burst Time: 20
Priority: 3
P[2]
Burst Time: 5
Priority: 4
P[3]


Activities
Places
Terminal
Thu Aug 31, 13:39
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc5
Burst Time: 10
Priority: 2
P[4]
Burst Time: 15
Priority: 1
Process Burst Time Waiting Time Turnaround Time
P[4] 15 0 15
P[3] 10 15 25
P[1] 20 25 45
P[2] 5 45 50
Average Waiting Time is: 21
Average Turnaround Time is: 33
[da_sh@localhost exc5]$ cc non-pre_sjf.c
[da_sh@localhost exc5]$ ./a.out
Enter the total number of processes: 4
Enter the Burst Time one by one:
p1:20
p2:10
p3:5
p4:15
Process Burst Time Waiting Time Turnaround Time
p3 5 0 5
p2 10 5 15
p4 15 15 30
p1 20 30 50
Average Waiting Time is: 12.500000
Average Turnaround Time is: 25.000000
[da_sh@localhost exc5]$

Exercise 6 – Preemptive Scheduling Algorithms

16BCE0783 – Daksh

Output Screenshots at Last

1. Pre Emptive Priority Scheduling Algorithm

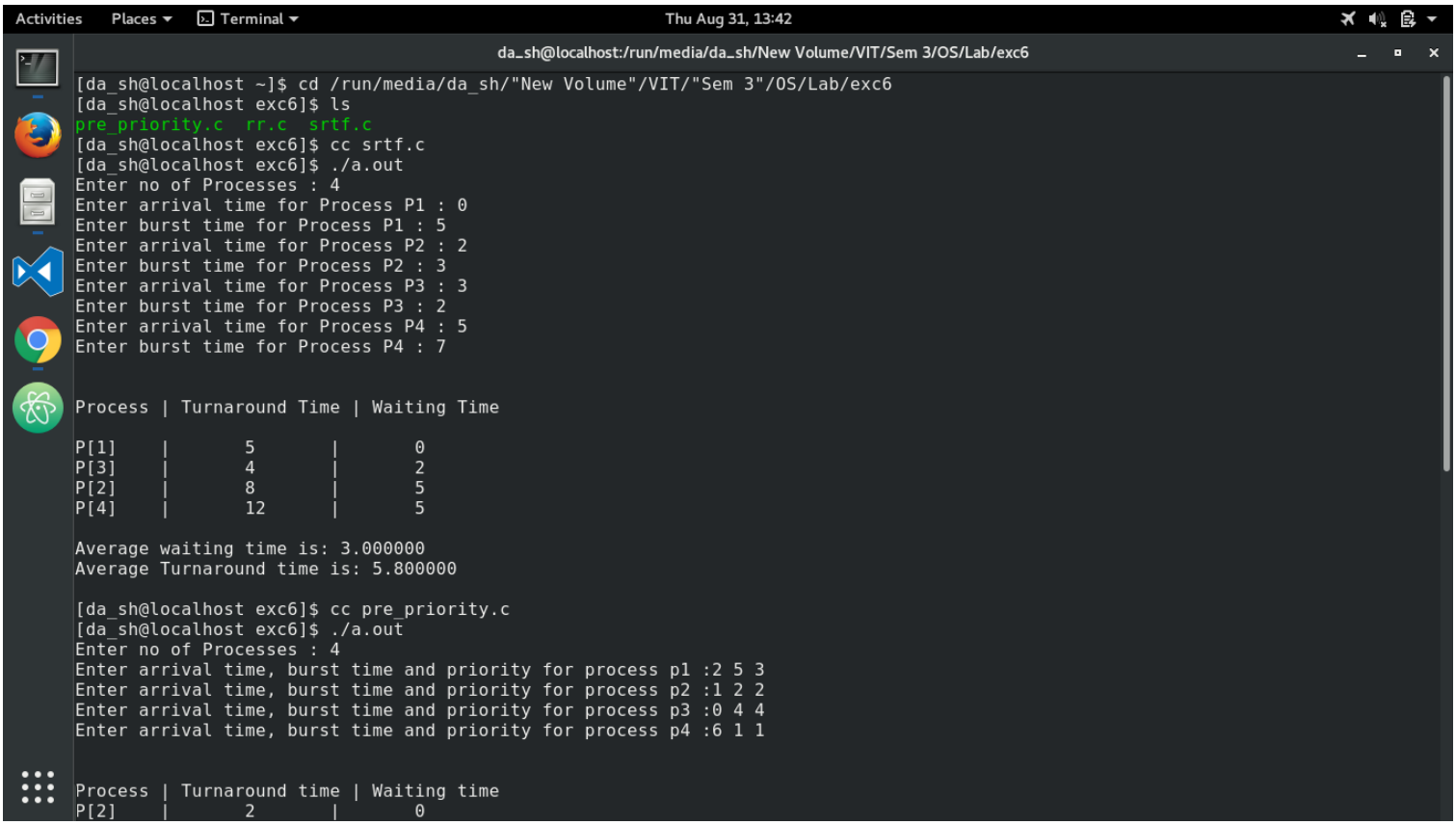
```
#include<stdio.h>

void main(){
  int n,time,wait_sum=0,j,turnaround_sum=0,i,smallest;
  int arrival_time[10],burst_time[10],priority[10],remain_time[10],remain;
  printf("Enter no of Processes : ");
  scanf("%d",&n);
  remain=n;
  for(i=0;i<n;i++){
    printf("Enter arrival time, burst time and priority for process p%d :",i+1);
    scanf("%d",&arrival_time[i]);
    scanf("%d",&burst_time[i]);
    scanf("%d",&priority[i]);
    remain_time[i]=burst_time[i];
  }
  priority[9]=11;
  printf("\n\nProcess\t| Turnaround time | Waiting time\n");
  for(time=0;remain!=0;time++){
    smallest=9;
    for(i=0;i<n;i++){
      if(arrival_time[i]<=time && priority[i]<priority[smallest] && remain_time[i]>0){
        smallest=i;
      }
    }
    remain_time[smallest]--;
    if(remain_time[smallest]==0){
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",smallest+1,time+1-arrival_time[smallest],time+1-
arrival_time[smallest]-burst_time[smallest]);
      wait_sum+=time+1-arrival_time[smallest];
      turnaround_sum+=time+1-arrival_time[smallest]-burst_time[smallest];
    }
  }
  printf("\nAvg waiting time is: %f\n",wait_sum*1.0/n);
  printf("Avg turnaround time is: %f\n\n",turnaround_sum*1.0/n);
}
```

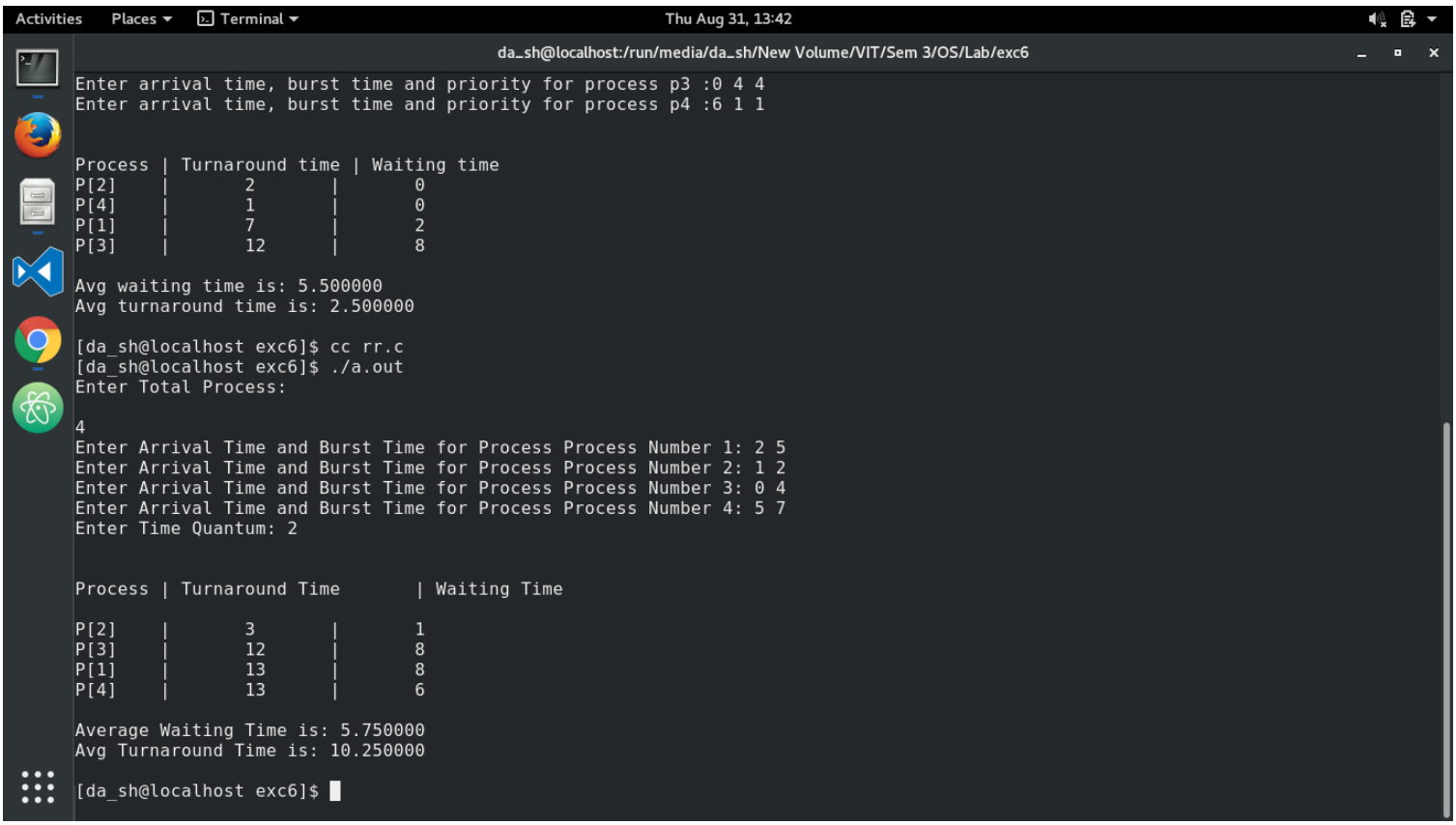
2. Round Robin Scheduling Algorithm

```
#include<stdio.h>

void main(){

    int j,time_quantum,time,n,remain,count,flag=0;
    int wait_time=0,turnaround_time=0,arrival_time[10],burst_time[10],r_var[10];
    printf("Enter Total Process: ");
    scanf("%d",&n);
    remain=n;
    for(count=0;count<n;count++) {
        printf("Enter Arrival Time and Burst Time for Process Process Number %d: ",count+1);
        scanf("%d",&arrival_time[count]);
        scanf("%d",&burst_time[count]);
        r_var[count]=burst_time[count];
    }
    printf("Enter Time Quantum: ");
    scanf("%d",&time_quantum);
    printf("\n\nProcess\t| Turnaround Time\t| Waiting Time\n\n");
    for(time=0,count=0;remain!=0;) {
        if(r_var[count]<=time_quantum && r_var[count]>0) {
            time+=r_var[count];
            r_var[count]=0;
            flag=1;
        }
        else if(r_var[count]>0) {
            r_var[count]-=time_quantum;
            time+=time_quantum;
        }
        if(r_var[count]==0 && flag==1) {
            remain--;
            printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-arrival_time[count],time-
arrival_time[count]-burst_time[count]);
            wait_time+=time-arrival_time[count]-burst_time[count];
            turnaround_time+=time-arrival_time[count];
            flag=0;
        }
        if(count==n-1)
            count=0;
        else if(arrival_time[count+1]<=time)
            count++;
        else
            count=0;
    }
    printf("\nAverage Waiting Time is: %f\n",wait_time*1.0/n);
    printf("Avg Turnaround Time is: %f\n\n",turnaround_time*1.0/n);
}
```

3. Shortest Remaining Time First (SRTF) Scheduling Algorithm

```c
#include<stdio.h>

void main(){
    int arrival_time[10],endTime,i,burst_time[10],remain_time[10],smallest;
    int remain=0,n,time,wait_sum=0,turnaround_sum=0;
    printf("Enter no of Processes : ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
        printf("Enter arrival time for Process P%d : ",i+1);
        scanf("%d",&arrival_time[i]);
        printf("Enter burst time for Process P%d : ",i+1);
        scanf("%d",&burst_time[i]);
        remain_time[i]=burst_time[i];
    }
    printf("\n\nProcess\t| Turnaround Time | Waiting Time\n");
    remain_time[9]=9999;
    for(time=0;remain!=n;time++){
        smallest=9;
        for(i=0;i<n;i++){
            if(arrival_time[i]<=time && remain_time[i]<remain_time[smallest] &&
remain_time[i]>0){
                smallest=i;
            }
        }
        remain_time[smallest]--;
        if(remain_time[smallest]==0){
            remain++;
            endTime=time+1;
            printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-arrival_time[smallest],endTime-
burst_time[smallest]-arrival_time[smallest]);
            wait_sum+=endTime-burst_time[smallest]-arrival_time[smallest];
            turnaround_sum+=endTime-arrival_time[smallest];
        }
    }
    printf("\n\nAverage waiting time is: %f\n",wait_sum*1.0/n);
    printf("Average Turnaround time is: %f\n\n",turnaround_sum*1.0/n);
}
```


Activities
Places
Terminal
Thu Aug 31, 13:42
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc6
[da_sh@localhost ~]$ cd /run/media/da_sh/"New Volume"/VIT/"Sem 3"/OS/Lab/exc6
[da_sh@localhost exc6]$ ls
pre_priority.c rr.c srtf.c
[da_sh@localhost exc6]$ cc srtf.c
[da_sh@localhost exc6]$ ./a.out
Enter no of Processes : 4
Enter arrival time for Process P1 : 0
Enter burst time for Process P1 : 5
Enter arrival time for Process P2 : 2
Enter burst time for Process P2 : 3
Enter arrival time for Process P3 : 3
Enter burst time for Process P3 : 2
Enter arrival time for Process P4 : 5
Enter burst time for Process P4 : 7
Process | Turnaround Time | Waiting Time
P[1] | 5 | 0
P[3] | 4 | 2
P[2] | 8 | 5
P[4] | 12 | 5
Average waiting time is: 3.000000
Average Turnaround time is: 5.800000
[da_sh@localhost exc6]$ cc pre_priority.c
[da_sh@localhost exc6]$ ./a.out
Enter no of Processes : 4
Enter arrival time, burst time and priority for process p1 :2 5 3
Enter arrival time, burst time and priority for process p2 :1 2 2
Enter arrival time, burst time and priority for process p3 :0 4 4
Enter arrival time, burst time and priority for process p4 :6 1 1
Process | Turnaround time | Waiting time
P[2] | 2 | 0


Activities
Places
Terminal
Thu Aug 31, 13:42
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc6
Enter arrival time, burst time and priority for process p3 :0 4 4
Enter arrival time, burst time and priority for process p4 :6 1 1
Process | Turnaround time | Waiting time
P[2] | 2 | 0
P[4] | 1 | 0
P[1] | 7 | 2
P[3] | 12 | 8
Avg waiting time is: 5.500000
Avg turnaround time is: 2.500000
[da_sh@localhost exc6]$ cc rr.c
[da_sh@localhost exc6]$ ./a.out
Enter Total Process:
4
Enter Arrival Time and Burst Time for Process Process Number 1: 2 5
Enter Arrival Time and Burst Time for Process Process Number 2: 1 2
Enter Arrival Time and Burst Time for Process Process Number 3: 0 4
Enter Arrival Time and Burst Time for Process Process Number 4: 5 7
Enter Time Quantum: 2
Process | Turnaround Time | Waiting Time
P[2] | 3 | 1
P[3] | 12 | 8
P[1] | 13 | 8
P[4] | 13 | 6
Average Waiting Time is: 5.750000
Avg Turnaround Time is: 10.250000
[da_sh@localhost exc6]$

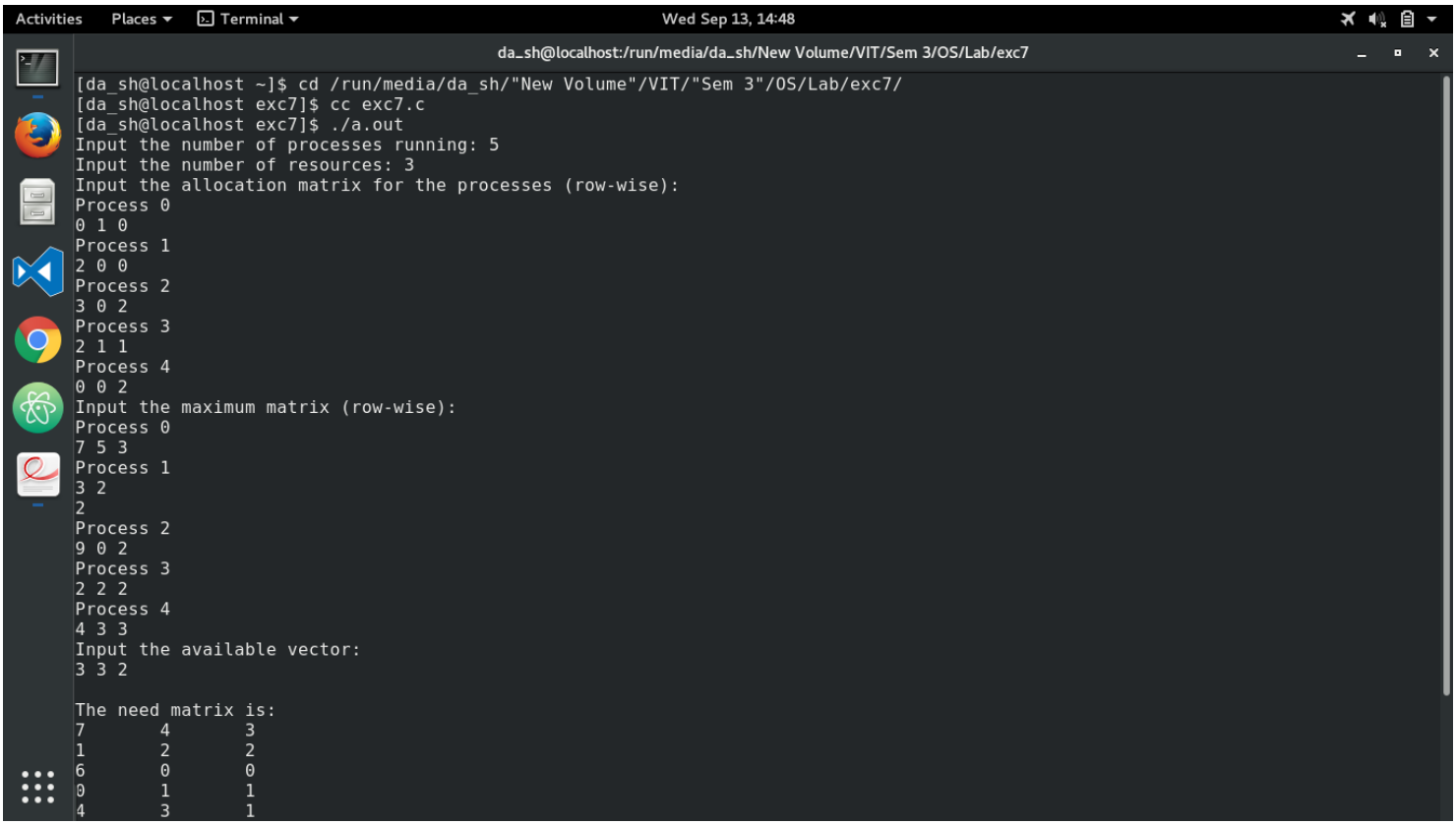
Exercise 7 – Banker's Algorithm

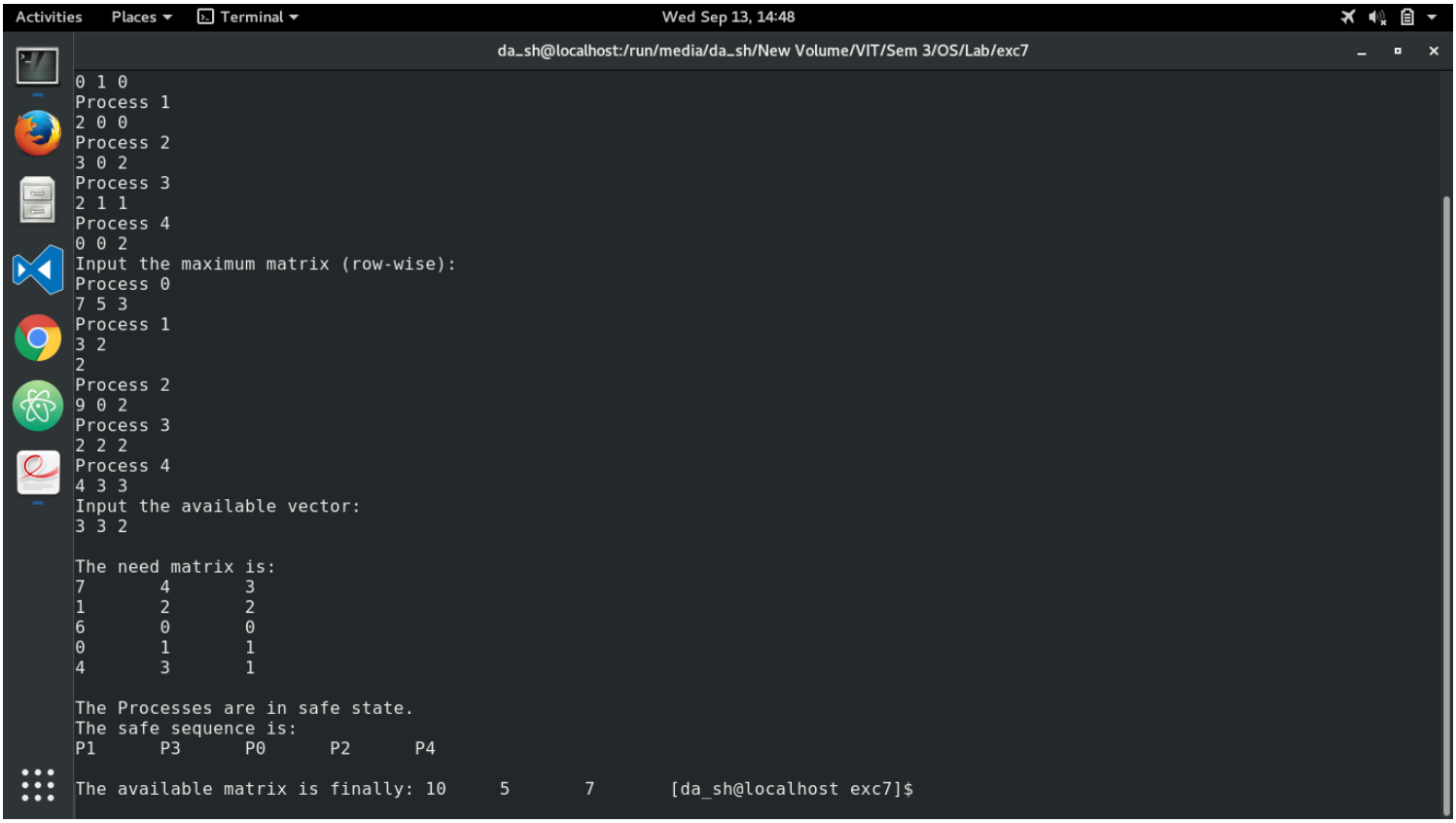
16BCE0783 – Daksh

Input Question is form theory class.

```
#include<stdio.h>

void main(){
    int allocation[10][5],max[10][5],need[10][5],available[3],flag[10],sq[10];
    int n,r,i,j,k,count,count1=0;
    printf("Input the number of processes running: ");
    scanf("%d",&n);
    for(i=0;i<10;i++)
        flag[i]=0;
    printf("Input the number of resources: ");
    scanf("%d",&r);
    printf("Input the allocation matrix for the processes (row-wise): \n");
    for(i=0;i<n;i++){
        printf("Process %d\n",i);
        for(j=0;j<r;j++){
            scanf("%d",&allocation[i][j]);
        }
    }
    printf("Input the maximum matrix (row-wise): \n");
    for(i=0;i<n;i++){
        printf("Process %d\n",i);
        for(j=0;j<r;j++){
            scanf("%d",&max[i][j]);
        }
    }
    printf("Input the available vector: \n");
    for(i=0;i<r;i++){
        scanf("%d",&available[i]);
    }
    printf("\nThe need matrix is: \n");
    for(i=0;i<n;i++){
        for(j=0;j<r;j++){
            need[i][j]= max[i][j]-allocation[i][j];
            printf("%d\t",need[i][j]);
        }
        printf("\n");
    }
    do{
    for(k=0;k<n;k++){
        for(i=0;i<n;i++){
            if(flag[i]==0){
                count=0;
```

```
                for(j=0;j<r;j++){
                    if(available[j]>=need[i][j])
                        count++;
                }
                if(count==r){
                    count1++;
                    flag[i]=1;
                    sq[count1-1]=i;
                    for(j=0;j<r;j++){
                        available[j]=available[j]+allocation[i][j];
                    }
                break;
                }
            }
        }
    }
    if(count1!=n){
        printf("\nThe Processes are in unsafe state.");
        break;
    }
    }while(count1!=n);
    if(count1==n){
        printf("\nThe Processes are in safe state.");
        printf("\nThe safe sequence is: \n");
        for(i=0;i<n;i++)
            printf("P%d\t",sq[i]);
        printf("\n");
        printf("\nThe available matrix is finally: ");
        for(i=0;i<r;i++)
            printf("%d\t",available[i]);
    }
}
```


Activities
Places
Terminal
Wed Sep 13, 14:48
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc7
[da_sh@localhost ~]$ cd /run/media/da_sh/"New Volume"/VIT/"Sem 3"/OS/Lab/exc7/
[da_sh@localhost exc7]$ cc exc7.c
[da_sh@localhost exc7]$ ./a.out
Input the number of processes running: 5
Input the number of resources: 3
Input the allocation matrix for the processes (row-wise):
Process 0
0 1 0
Process 1
2 0 0
Process 2
3 0 2
Process 3
2 1 1
Process 4
0 0 2
Input the maximum matrix (row-wise):
Process 0
7 5 3
Process 1
3 2
2
Process 2
9 0 2
Process 3
2 2 2
Process 4
4 3 3
Input the available vector:
3 3 2
The need matrix is:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1


Activities
Places
Terminal
Wed Sep 13, 14:48
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc7
0 1 0
Process 1
2 0 0
Process 2
3 0 2
Process 3
2 1 1
Process 4
0 0 2
Input the maximum matrix (row-wise):
Process 0
7 5 3
Process 1
3 2
2
Process 2
9 0 2
Process 3
2 2 2
Process 4
4 3 3
Input the available vector:
3 3 2
The need matrix is:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
The Processes are in safe state.
The safe sequence is:
P1 P3 P0 P2 P4
The available matrix is finally: 10 5 7 [da_sh@localhost exc7]$

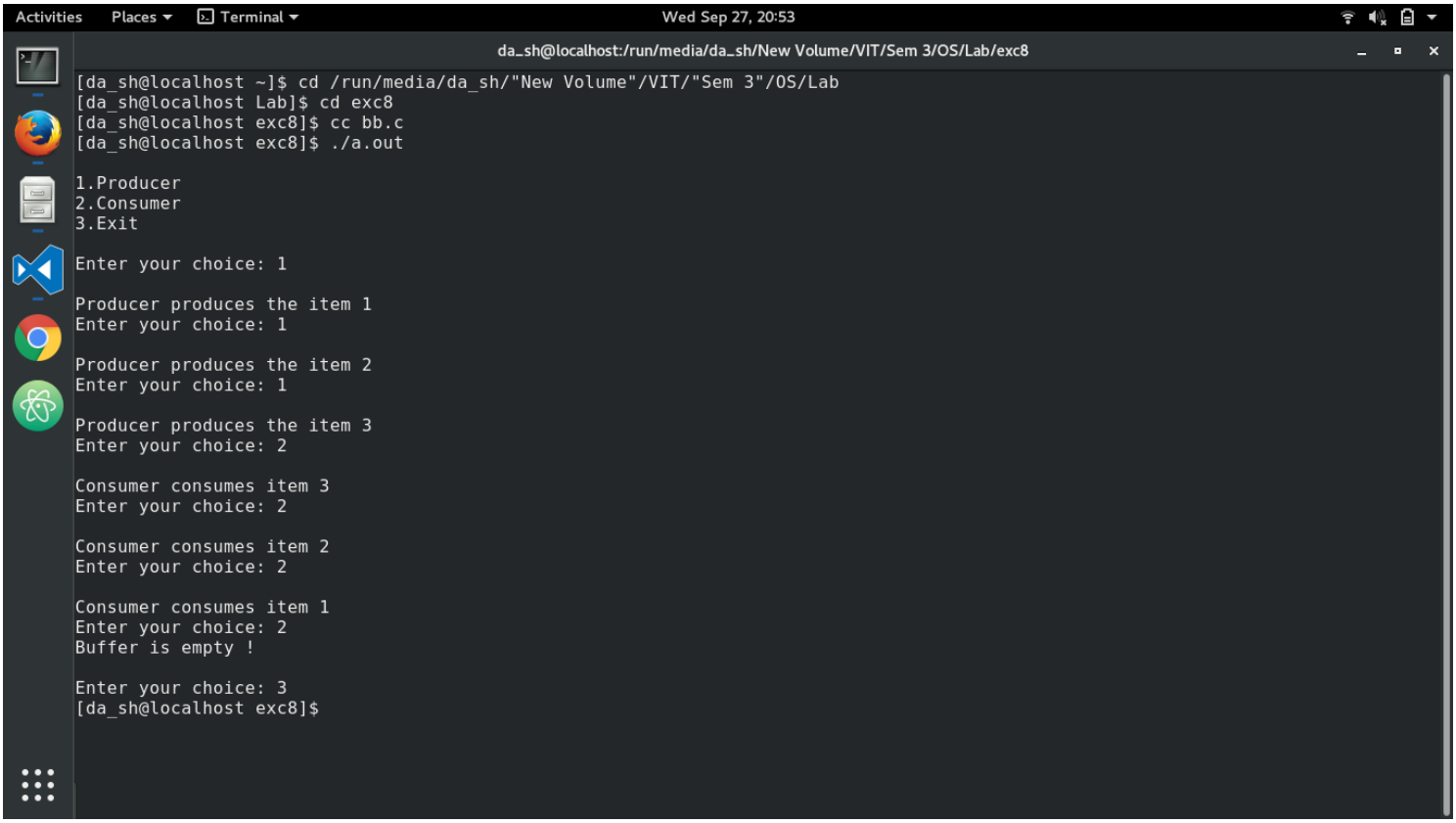
Exercise 8 – Classical Problems of Synchronization

16BCE0783 – Daksh

1. Bounded Buffer / Producer Consumer Problem Algorithm

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

int mutex=1,full=0,empty=3,x=0;

void main(){
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.Producer\n2.Consumer\n3.Exit\n");
    while(1){
        printf("\nEnter your choice: ");
        scanf("%d",&n);
        switch(n){
            case 1:    if((mutex==1)&&(empty!=0))
                           producer();
                       else
                           printf("Buffer is full ! \n");
                       break;
            case 2:    if((mutex==1)&&(full!=0))
                           consumer();
                       else
                           printf("Buffer is empty ! \n");
                       break;
            case 3:
                       exit(0);
                       break;
        }
    }
}

int wait(int s){
    return (--s);
}

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

void producer(){
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nProducer produces the item %d",x);
    mutex=signal(mutex);
}
```

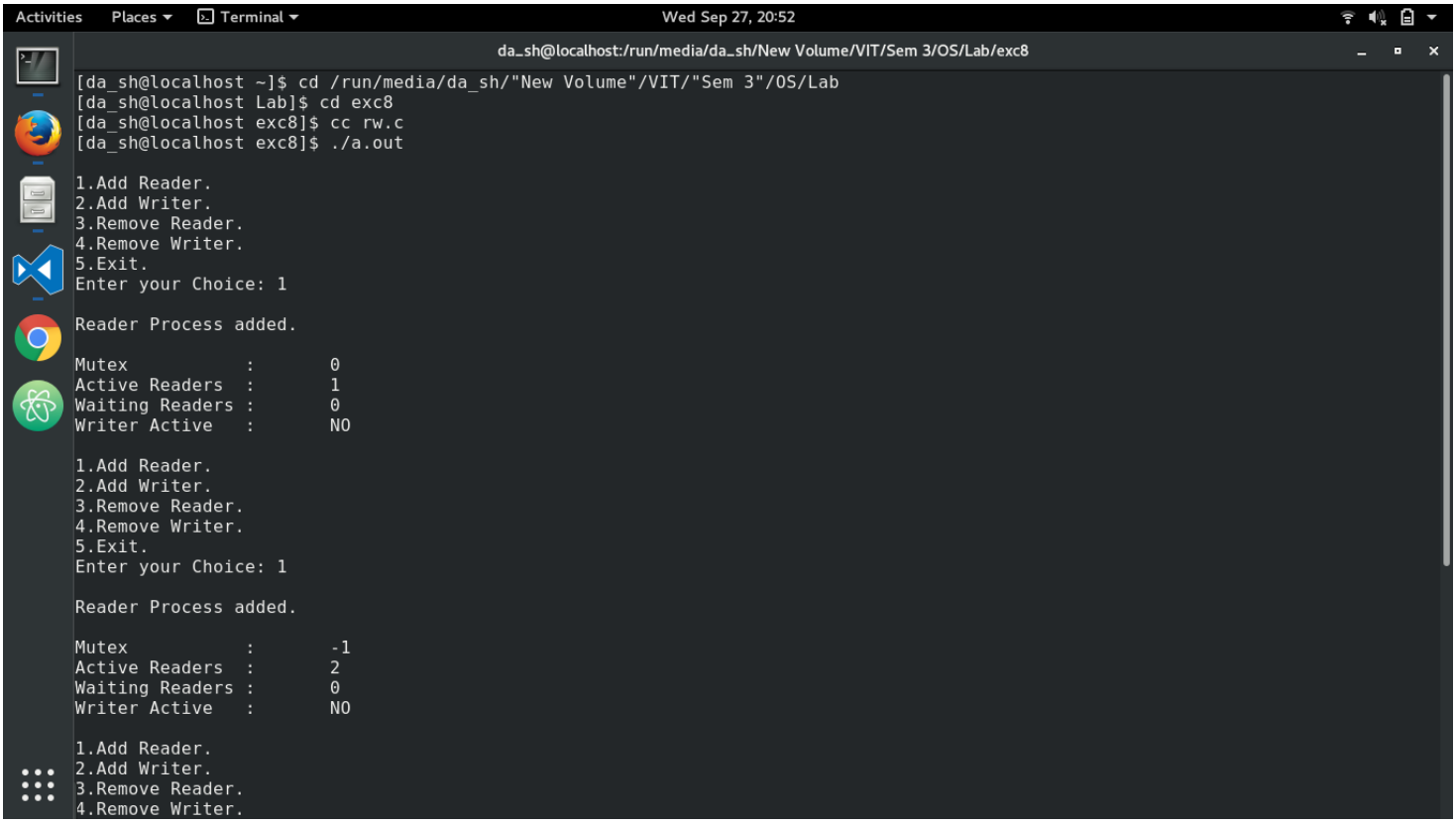
```
void consumer(){
    mutex=wait(mutex);
    full=wait(full);
    empty=signal(empty);
    printf("\nConsumer consumes item %d",x);
    x--;
    mutex=signal(mutex);
}
```

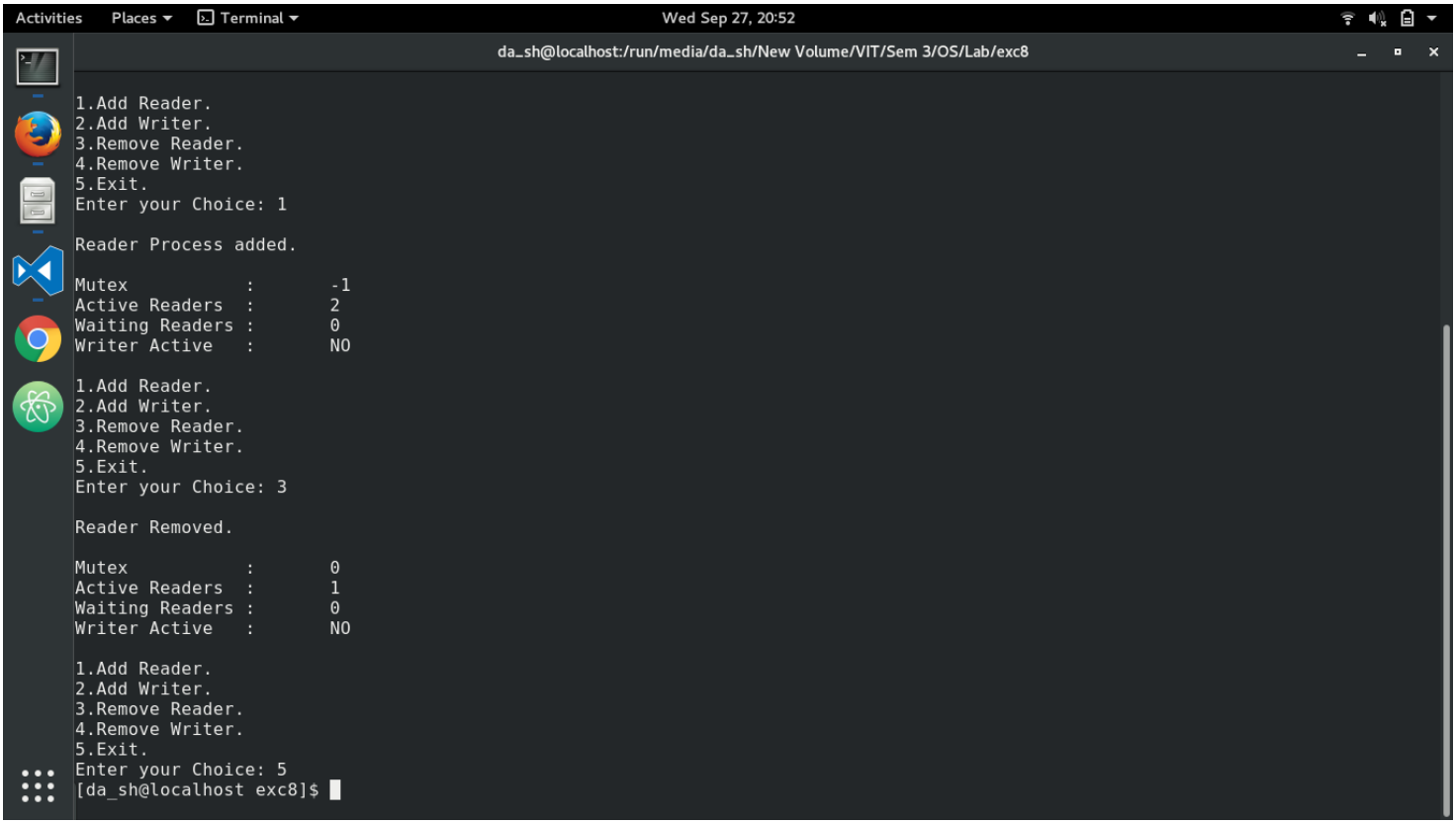


2. Readers Writers Problem Algorithm

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

struct semaphore{
    int mutex;
    int rcount;
    int rwait;
    bool wrt;
};

void addR(struct semaphore *s){
    if (s->mutex == 0 && s->rcount == 0){
        printf("\nSorry, File open in Write mode.\nNew Reader added to queue.\n");
        s->rwait++;
    }
    else{
        printf("\nReader Process added.\n");
        s->rcount++;
        s->mutex--;
    }
}

void addW(struct semaphore *s){
    if(s->mutex==1){
        s->mutex--;
        s->wrt=1;
        printf("\nWriter Process added.\n");
    }
    else if(s->wrt)
        printf("\nSorry, Writer already operational.\n");
    else
        printf("\nSorry, File open in Read mode.\n");
}

void remR(struct semaphore *s){
    if(s->rcount == 0)
        printf("\nNo readers to remove.\n");
    else{
        printf("\nReader Removed.\n");
        s->rcount--;
        s->mutex++;
    }
}

void remW(struct semaphore *s){
    if(s->wrt==0)
        printf("\nNo Writer to Remove");
    else{
        printf("\nWriter Removed\n");
        s->mutex++;
        s->wrt=0;
        if(s->rwait!=0){
            s->mutex-=s->rwait;
            s->rcount=s->rwait;
            s->rwait=0;
            printf("%d waiting Readers Added.",s->rcount);
        }
```

```
    }
}

void main(){
    struct semaphore S1={1,0,0};
    while(1){
        printf("\n1.Add Reader.\n2.Add Writer.\n3.Remove Reader.\n4.Remove
Writer.\n5.Exit.\nEnter your Choice: ");
        int ch;
        scanf("%d",&ch);
        switch(ch){
            case 1: addR(&S1); break;
            case 2: addW(&S1); break;
            case 3: remR(&S1); break;
            case 4: remW(&S1); break;
            case 5: exit(0);
        }
        printf("\nMutex\t\t:\t%d\nActive Readers\t:\t%d\nWaiting
Readers\t:\t%d\nWriter Active\t:\t%s\n", S1.mutex, S1.rcount, S1.rwait, (S1.mutex==0
&& S1.rcount==0) ? "YES" : "NO");
    }
}
```



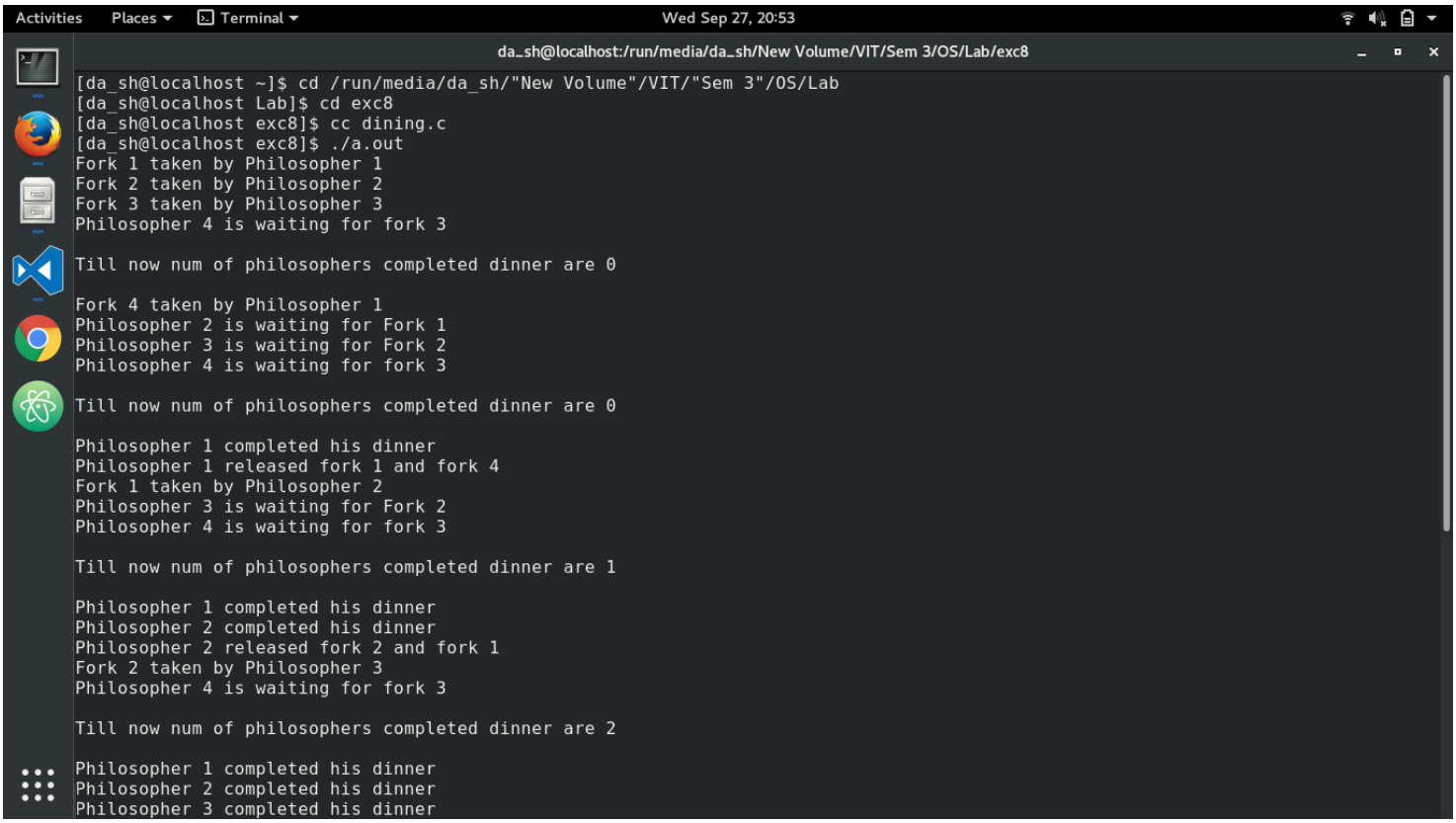


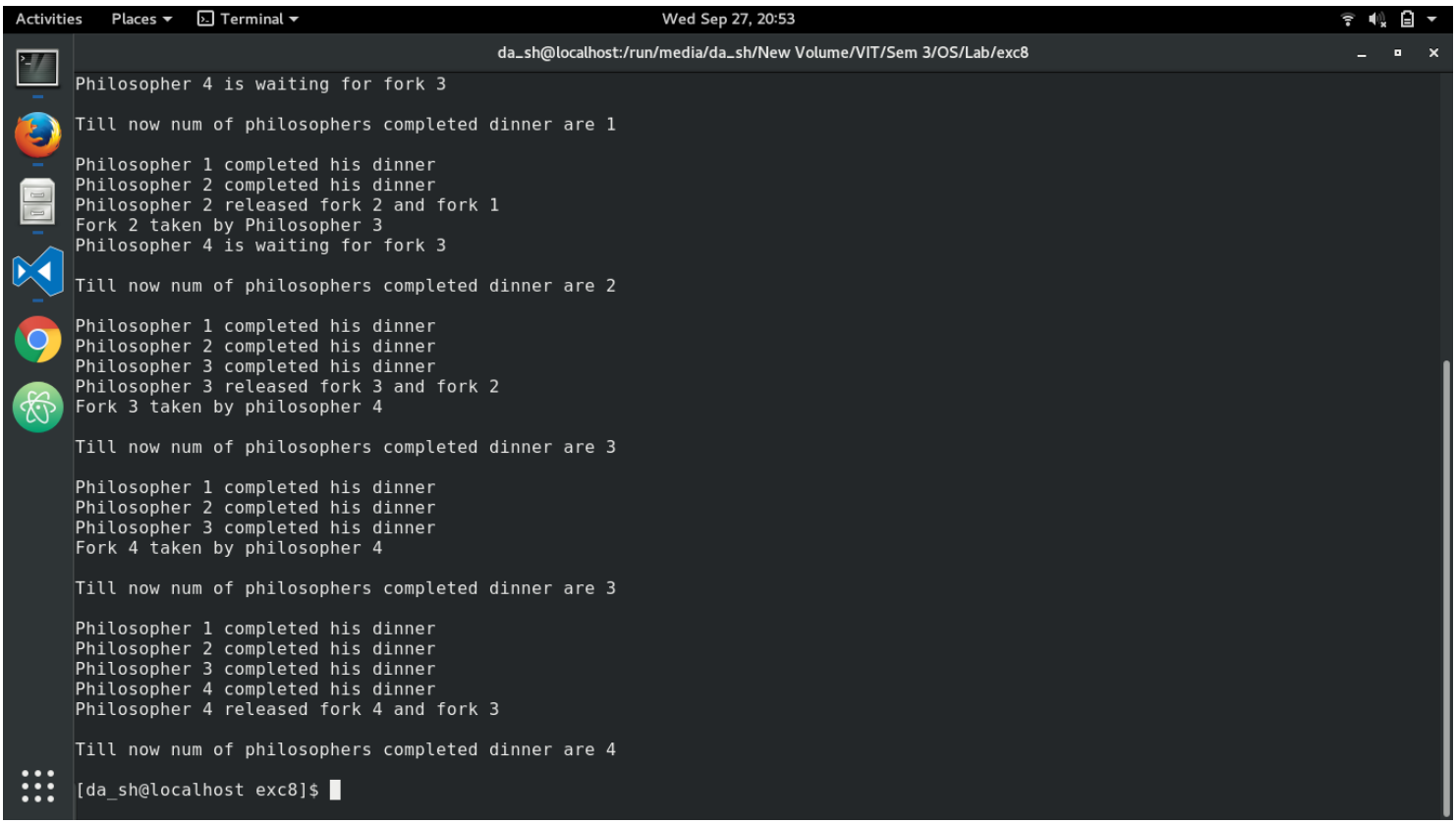
3. Dining Philosopher's Problem Algorithm

```
#include<stdio.h>
#define n 4

int compltedPhilo = 0,i;

struct fork{
   int taken;
}ForkAvil[n];

struct philosp{
   int left;
   int right;
}Philostatus[n];

void goForDinner(int philID){
    if(Philostatus[philID].left==10 && Philostatus[philID].right==10)
        printf("Philosopher %d completed his dinner\n",philID+1);
    else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){
        printf("Philosopher %d completed his dinner\n",philID+1);

        Philostatus[philID].left = Philostatus[philID].right = 10;
        int otherFork = philID-1;

        if(otherFork== -1)
            otherFork=(n-1);

        ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;
        printf("Philosopher %d released fork %d and fork
%d\n",philID+1,philID+1,otherFork+1);
```

```
        compltedPhilo++;
    }
    else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){
        if(philID==(n-1)){
            if(ForkAvil[philID].taken==0){
                ForkAvil[philID].taken = Philostatus[philID].right = 1;
                printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);
            }
            else{
                printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);
            }
        }
        else{
            int dupphilID = philID;
            philID-=1;

            if(philID== -1)
                philID=(n-1);

            if(ForkAvil[philID].taken == 0){
                ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);
            }
            else{
                printf("Philosopher %d is waiting for Fork
%d\n",dupphilID+1,philID+1);
            }
        }
    }
    else if(Philostatus[philID].left==0){
        if(philID==(n-1)){
            if(ForkAvil[philID-1].taken==0){
                ForkAvil[philID-1].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by philosopher %d\n",philID,philID+1);
            }
            else{
                printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);
            }
        }
        else{
            if(ForkAvil[philID].taken == 0){
                ForkAvil[philID].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);
            }
            else{
                printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);
            }
        }
    }
}

void main(){
    for(i=0;i<n;i++)
        ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;

    while(compltedPhilo<n){
        for(i=0;i<n;i++)
           goForDinner(i);
        printf("\nTill now num of philosophers completed dinner are
%d\n\n",compltedPhilo);
```

```
    }
}
```


Activities
Places
Terminal
Wed Sep 27, 20:53
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc8
[da_sh@localhost ~]$ cd /run/media/da_sh/"New Volume"/VIT/"Sem 3"/OS/Lab
[da_sh@localhost Lab]$ cd exc8
[da_sh@localhost exc8]$ cc dining.c
[da_sh@localhost exc8]$ ./a.out
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 0
Fork 4 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 0
Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 4
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 1
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 2
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner

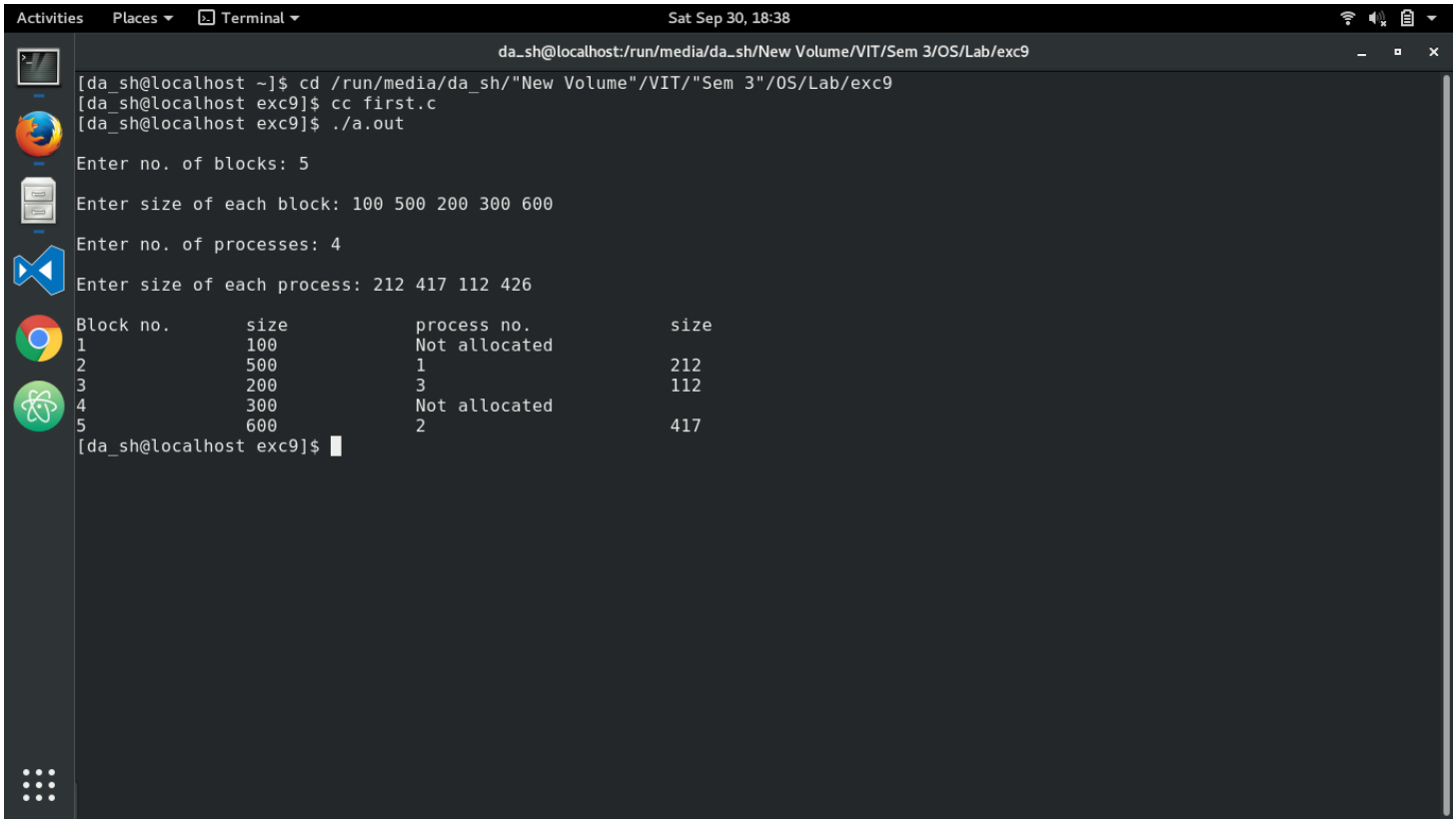

Activities
Places
Terminal
Wed Sep 27, 20:53
da_sh@localhost:/run/media/da_sh/New Volume/VIT/Sem 3/OS/Lab/exc8
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 1
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3
Till now num of philosophers completed dinner are 2
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4
Till now num of philosophers completed dinner are 3
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by philosopher 4
Till now num of philosophers completed dinner are 3
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3
Till now num of philosophers completed dinner are 4
[da_sh@localhost exc8]$

Exercise 9 – Memory Management Algorithms

16BCE0783 – Daksh

1. First Fit Algorithm

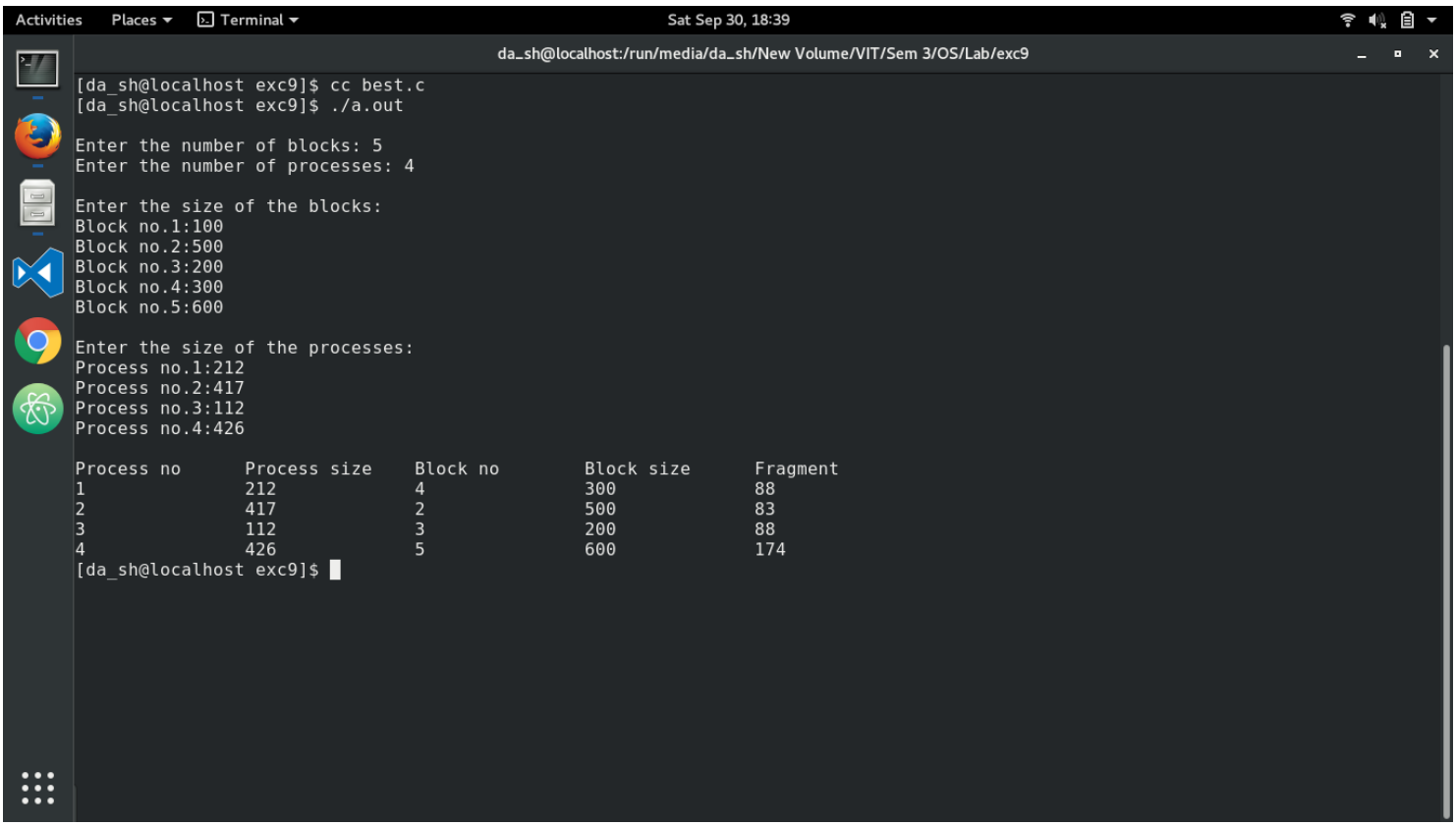
```
#include<stdio.h>

void main(){
   int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
   for(i = 0; i < 10; i++){
       flags[i] = 0;
       allocation[i] = -1;
   }

   printf("\nEnter no. of blocks: ");
   scanf("%d", &bno);

   printf("\nEnter size of each block: ");
   for(i = 0; i < bno; i++)
       scanf("%d", &bsize[i]);

   printf("\nEnter no. of processes: ");
   scanf("%d", &pno);

   printf("\nEnter size of each process: ");
   for(i = 0; i < pno; i++)
       scanf("%d", &psize[i]);
   for(i = 0; i < pno; i++){
       for(j = 0; j < bno; j++){
           if(flags[j] == 0 && bsize[j] >= psize[i]){
               allocation[j] = i;
               flags[j] = 1;
               break;
           }
        }
    }

   printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
   for(i = 0; i < bno; i++){
       printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
       if(flags[i] == 1)
           printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);
       else
           printf("Not allocated");
   }
   printf("\n");
}
```



2. Best Fit Algorithm

```
#include<stdio.h>

void main(){
   int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;
   static int barray[20],parray[20];

   printf("\nEnter the number of blocks: ");
   scanf("%d",&nb);
   printf("Enter the number of processes: ");
   scanf("%d",&np);

   printf("\nEnter the size of the blocks: \n");
   for(i=1;i<=nb;i++){
       printf("Block no.%d:",i);
       scanf("%d",&b[i]);
   }

   printf("\nEnter the size of the processes: \n");
   for(i=1;i<=np;i++){
       printf("Process no.%d:",i);
       scanf("%d",&p[i]);
   }

   for(i=1;i<=np;i++){
       for(j=1;j<=nb;j++){
           if(barray[j]!=1){
               temp=b[j]-p[i];
               if(temp>=0){
                   if(lowest>temp){
```

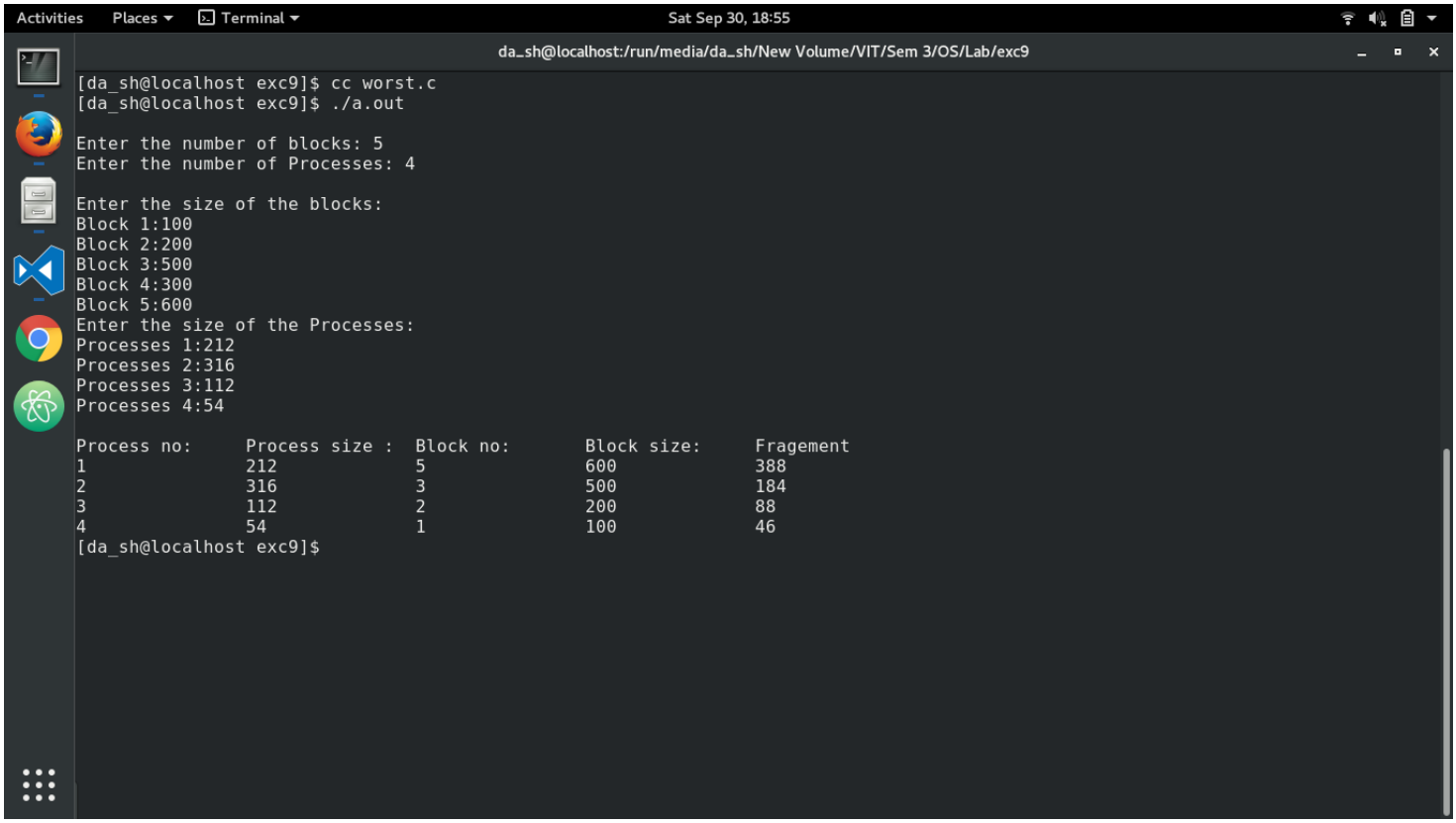
```
                        parray[i]=j;
                        lowest=temp;
                    }
                }
            }
        }

        fragment[i]=lowest;
        barray[parray[i]]=1;
        lowest=10000;
    }

   printf("\nProcess no\tProcess size\tBlock no\tBlock size\tFragment");
   for(i=1;i<=np && parray[i]!=0;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]);
    printf("\n");
}
```



3. Worst Fit Algorithm

```
#include<stdio.h>
#define max 25

void main(){
    int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
    int bf[max],ff[max];
    printf("\nEnter the number of blocks: ");
    scanf("%d",&nb);
    printf("Enter the number of Processes: ");
```

```
    scanf("%d",&nf);
    printf("\nEnter the size of the blocks: \n");
    for(i=1;i<=nb;i++){
        printf("Block %d:",i);
        scanf("%d",&b[i]);
    }
    printf("Enter the size of the Processes: \n");
    for(i=1;i<=nf;i++){
        printf("Processes %d:",i);
        scanf("%d",&f[i]);
    }
    for(i=1;i<=nf;i++){
        for(j=1;j<=nb;j++){
            if(bf[j]!=1){
                temp=b[j]-f[i];
                if(temp>=0){
                    if(highest<temp){
                        ff[i]=j;
                        highest=temp;
                    }
                }
            }
        }
        frag[i]=highest;
        bf[ff[i]]=1;
        highest=0;
    }
    printf("\nProcess no:\tProcess size :\tBlock no:\tBlock size:\tFragement");
    for(i=1;i<=nf;i++){
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
    }
    printf("\n");
}
```

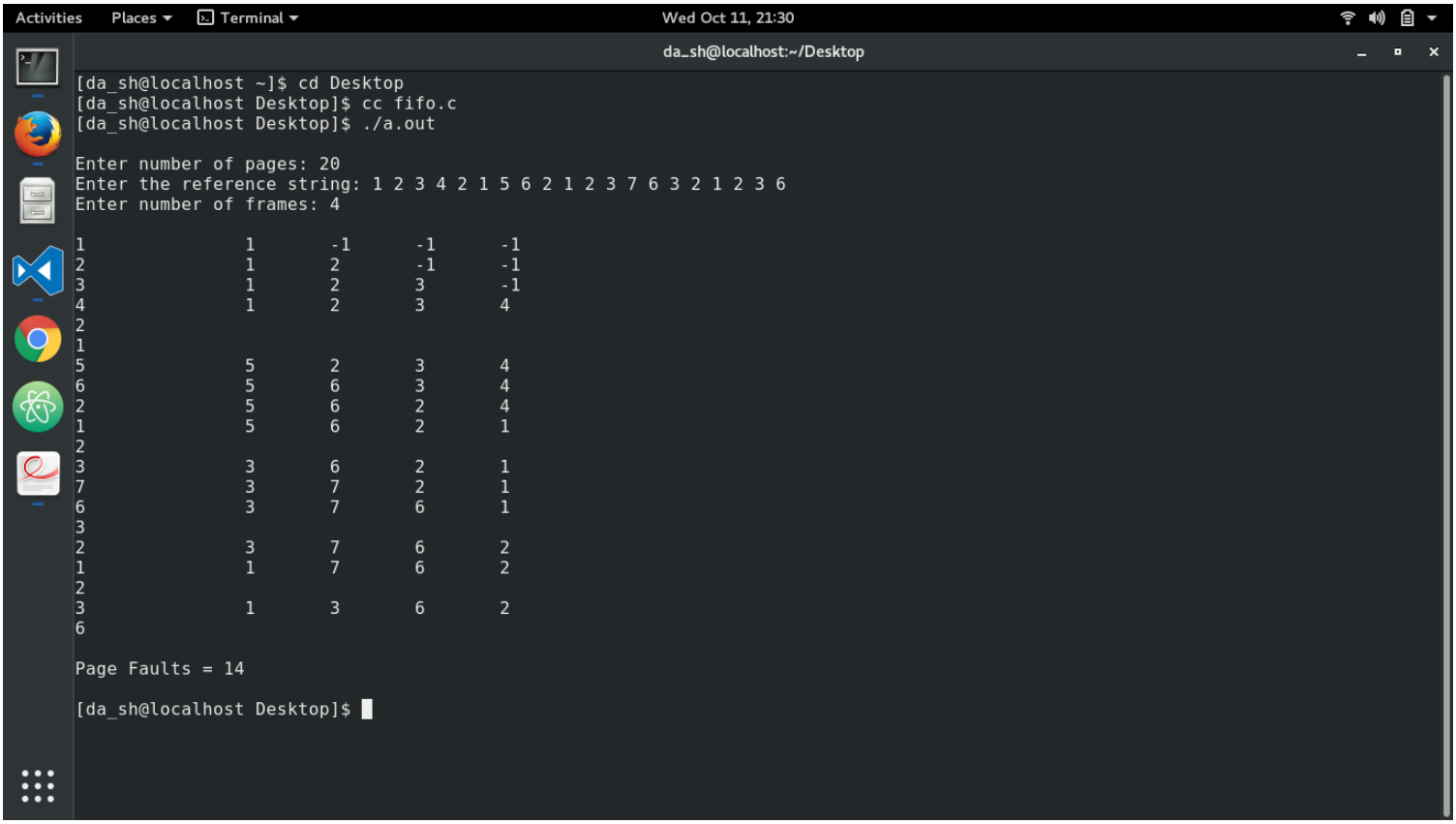


Exercise 10 – Page Replacement Algorithms

16BCE0783 – Daksh

1. FIFO Page Replacement Algorithm

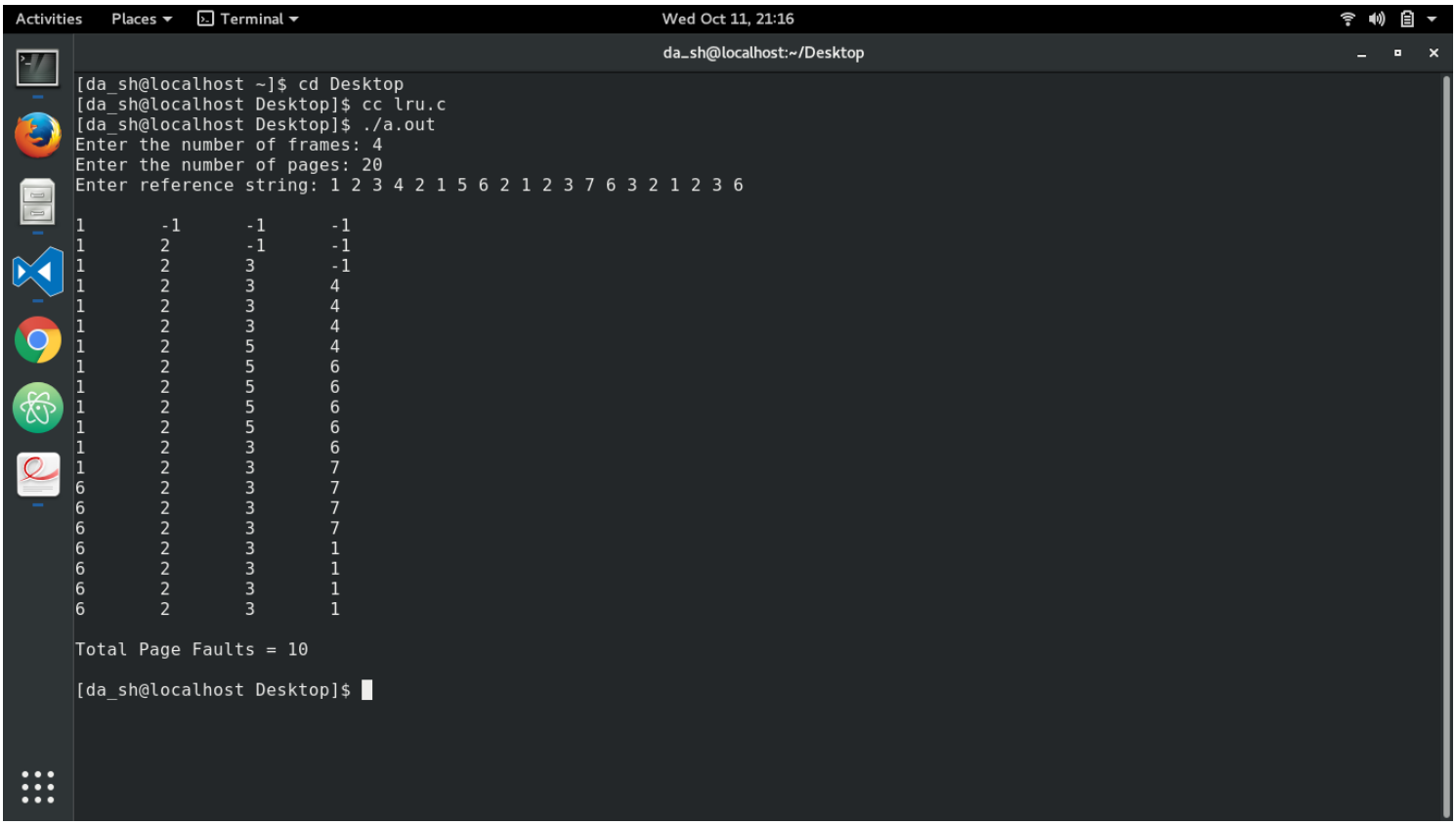
```
#include<stdio.h>
void main(){
	int i,j,n,a[50],frame[10],no,k,avail,count=0;
	printf("\nEnter number of pages: ");
	scanf("%d",&n);
	printf("Enter the reference string: ");
	for(i=1;i<=n;i++)
		scanf("%d",&a[i]);
	printf("Enter number of frames: ");
	scanf("%d",&no);
	printf("\n");
	for(i=0;i<no;i++)
		frame[i]= -1;
	j=0;
	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("\nPage Faults = %d\n\n",count);
}
```



2. LRU Page Replacement Algorithm

```
#include<stdio.h>

int calculate(int time[], int n){
    int i, minval = time[0], pos = 0;
    for(i = 1; i < n; ++i){
        if(time[i] < minval){
            minval = time[i];
            pos = i;
        }
    }
    return pos;
}

void main(){
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10],
flag1, flag2, i, j, pos, faults = 0;
    printf("Enter the number of frames: ");
    scanf("%d", &no_of_frames);
    printf("Enter the number of pages: ");
    scanf("%d", &no_of_pages);
    printf("Enter 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){
```

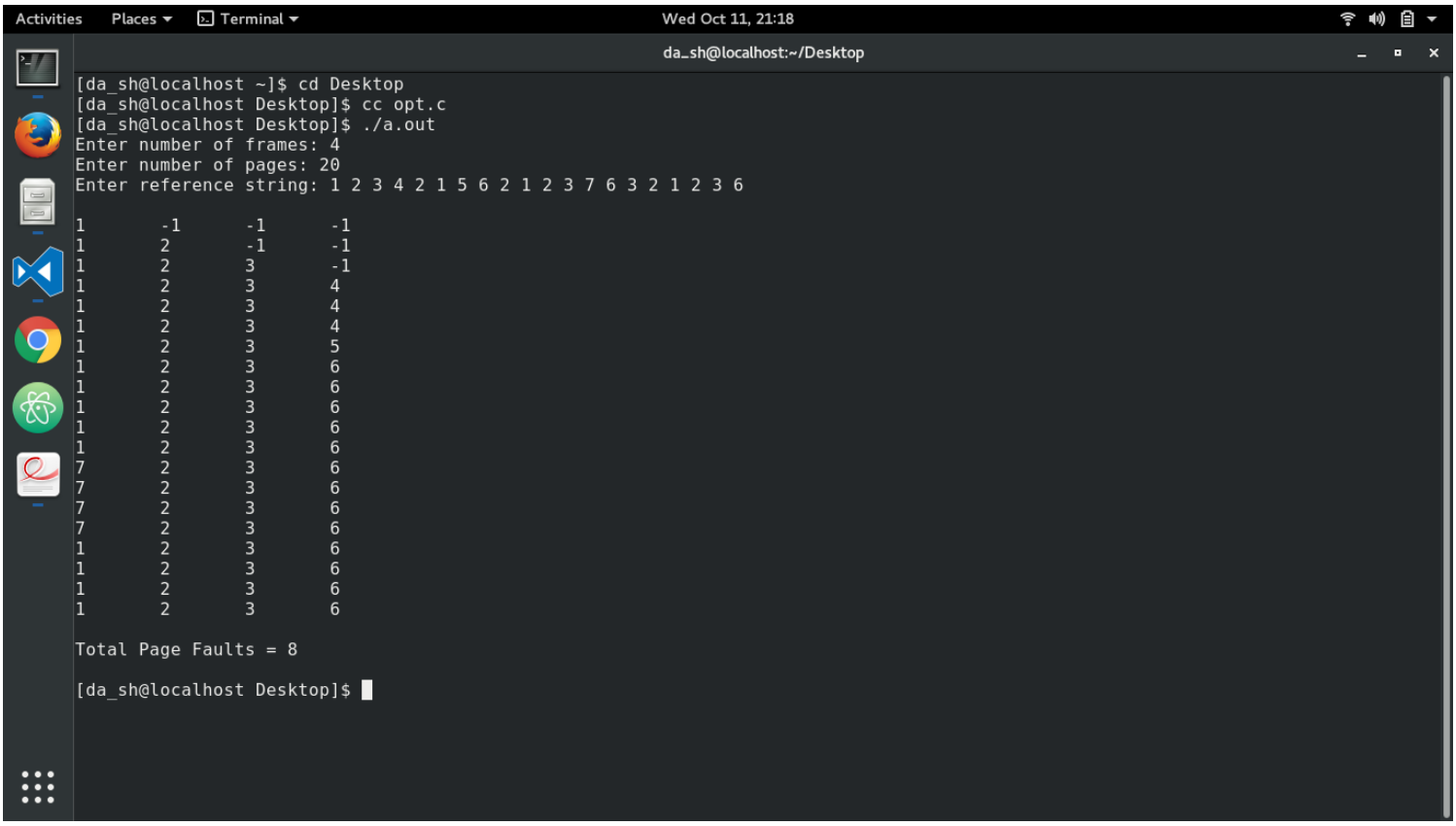
```
        flag1 = flag2 = 0;
        for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
                counter++;
                time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
        }
        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
        }
        if(flag2 == 0){
            pos = calculate(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
        }
        printf("\n");
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }
    printf("\n\nTotal Page Faults = %d\n\n", faults);
}
```



3. Optimal Page Replacement Algorithm

```
#include<stdio.h>

void main(){
    int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2,
flag3, i, 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 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\n\n", faults);
}
```


Activities
Places
Terminal
Wed Oct 11, 21:18
da_sh@localhost:~/Desktop
[da_sh@localhost ~]$ cd Desktop
[da_sh@localhost Desktop]$ cc opt.c
[da_sh@localhost Desktop]$ ./a.out
Enter number of frames: 4
Enter number of pages: 20
Enter reference string: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1 -1 -1 -1
1 2 -1 -1
1 2 3 -1
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 5
1 2 3 6
1 2 3 6
1 2 3 6
1 2 3 6
1 2 3 6
7 2 3 6
7 2 3 6
7 2 3 6
7 2 3 6
1 2 3 6
1 2 3 6
1 2 3 6
1 2 3 6
Total Page Faults = 8
[da_sh@localhost Desktop]$