|  |  |  |  |
| --- | --- | --- | --- |
|  | **COLLEGE OF COMPUTING AND INFORMATION SCIENCES** | | |
| **Assignment # 02** | | |
| **Course Title** | Operating System | **Total Marks** | 10 |
| **Date** | 18/04/2022 | **Class ID** | 109134 |
| **Student Id** | **11067** | **Student Name** | **SYED MUZZAMIL WASEEM** |

**Instructions:**

* Copied work and late submission will be marked as ZERO.
* Attach your code and screenshot of your output in this file.
* Submit hardcopy of your solution in class.

**Submission Deadline: 18-04-2022**

**Question 1:**

Consider the following scenario. There is a ticket booking counter that sells or cancels tickets for a

train berth. Initially, there are a total of 10 berths available numbered from 101-110.

Only one person can buy or cancel a ticket at a time. A person gets the first berth available from

the numbered berths. If a person cancels a bought ticket, that berth will be made available. Every

person who buys a ticket gets a ticket number and the booked berth number. The first ticket is

numbered 1001 and for every successful buy, the number increases by 1, continuously.

Implement a program (write two different functions ticket\_buy() and ticket\_cancel() to be called

from main() ) for this scenario when there are 20 people (1 to 20) who are standing in a queue in

any order to buy a ticket. 10 of these persons from the queue are initially successful in getting a

ticket (ticket number 1001 to 1010). Then tickets for berth number 101, 102, and 105 are

cancelled, so three next persons from the queue will get the tickets (number 1011 to 1013).

Implement while considering what happens in real life scenario when multiple people want to buy

a ticket at the same time and how it is handled.

Consider the following scenario. There is a ticket booking counter that sells or cancels tickets for a

train berth. Initially, there are a total of 10 berths available numbered from 101-110.

Only one person can buy or cancel a ticket at a time. A person gets the first berth available from

the numbered berths. If a person cancels a bought ticket, that berth will be made available. Every

person who buys a ticket gets a ticket number and the booked berth number. The first ticket is

numbered 1001 and for every successful buy, the number increases by 1, continuously.

Implement a program (write two different functions ticket\_buy() and ticket\_cancel() to be called

from main() ) for this scenario when there are 20 people (1 to 20) who are standing in a queue in

any order to buy a ticket. 10 of these persons from the queue are initially successful in getting a

ticket (ticket number 1001 to 1010). Then tickets for berth number 101, 102, and 105 are

cancelled, so three next persons from the queue will get the tickets (number 1011 to 1013).

Implement while considering what happens in real life scenario when multiple people want to buy

a ticket at the same time and how it is handled.

Consider the following scenario. There is a ticket booking counter that sells or cancels tickets for a

train berth. Initially, there are a total of 10 berths available numbered from 101-110.

Only one person can buy or cancel a ticket at a time. A person gets the first berth available from

the numbered berths. If a person cancels a bought ticket, that berth will be made available. Every

person who buys a ticket gets a ticket number and the booked berth number. The first ticket is

numbered 1001 and for every successful buy, the number increases by 1, continuously.

Implement a program (write two different functions ticket\_buy() and ticket\_cancel() to be called

from main() ) for this scenario when there are 20 people (1 to 20) who are standing in a queue in

any order to buy a ticket. 10 of these persons from the queue are initially successful in getting a

ticket (ticket number 1001 to 1010). Then tickets for berth number 101, 102, and 105 are

cancelled, so three next persons from the queue will get the tickets (number 1011 to 1013).

Implement while considering what happens in real life scenario when multiple people want to buy

a ticket at the same time and how it is handled.

Consider the following scenario. There is a ticket booking counter that sells or cancels tickets for a

train berth. Initially, there are a total of 10 berths available numbered from 101-110.

Only one person can buy or cancel a ticket at a time. A person gets the first berth available from

the numbered berths. If a person cancels a bought ticket, that berth will be made available. Every

person who buys a ticket gets a ticket number and the booked berth number. The first ticket is

numbered 1001 and for every successful buy, the number increases by 1, continuously.

Implement a program (write two different functions ticket\_buy() and ticket\_cancel() to be called

from main() ) for this scenario when there are 20 people (1 to 20) who are standing in a queue in

any order to buy a ticket. 10 of these persons from the queue are initially successful in getting a

ticket (ticket number 1001 to 1010). Then tickets for berth number 101, 102, and 105 are

cancelled, so three next persons from the queue will get the tickets (number 1011 to 1013).

Implement while considering what happens in real life scenario when multiple people want to buy

a ticket at the same time and how it is handled.

Write down the following programs using shell script:

1. Write a shell script program for comparison of strings.

CODE:

#!/bin/bash

read -p "Enter first string: " msg1 read -p "Enter second string: " msg2

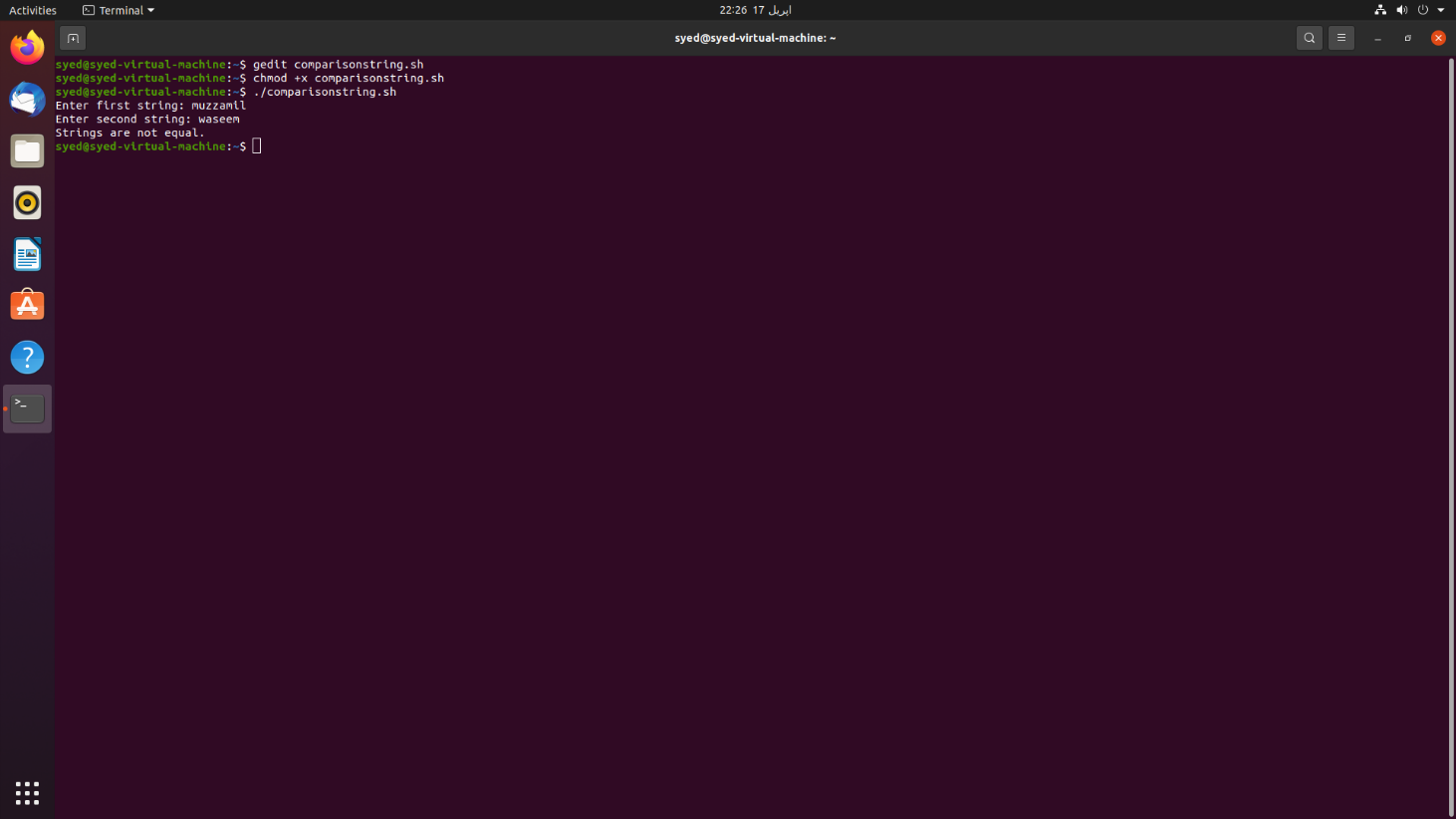
if [ "$msg1" == "$msg2" ]; then echo "Strings are equal."

else

echo "Strings are not equal."

fi

SCREENSHOT:



1. Calculate the factorial value of a number using shell script.

CODE:

#!/bin/bash

echo "Enter any number"

read number

f=1

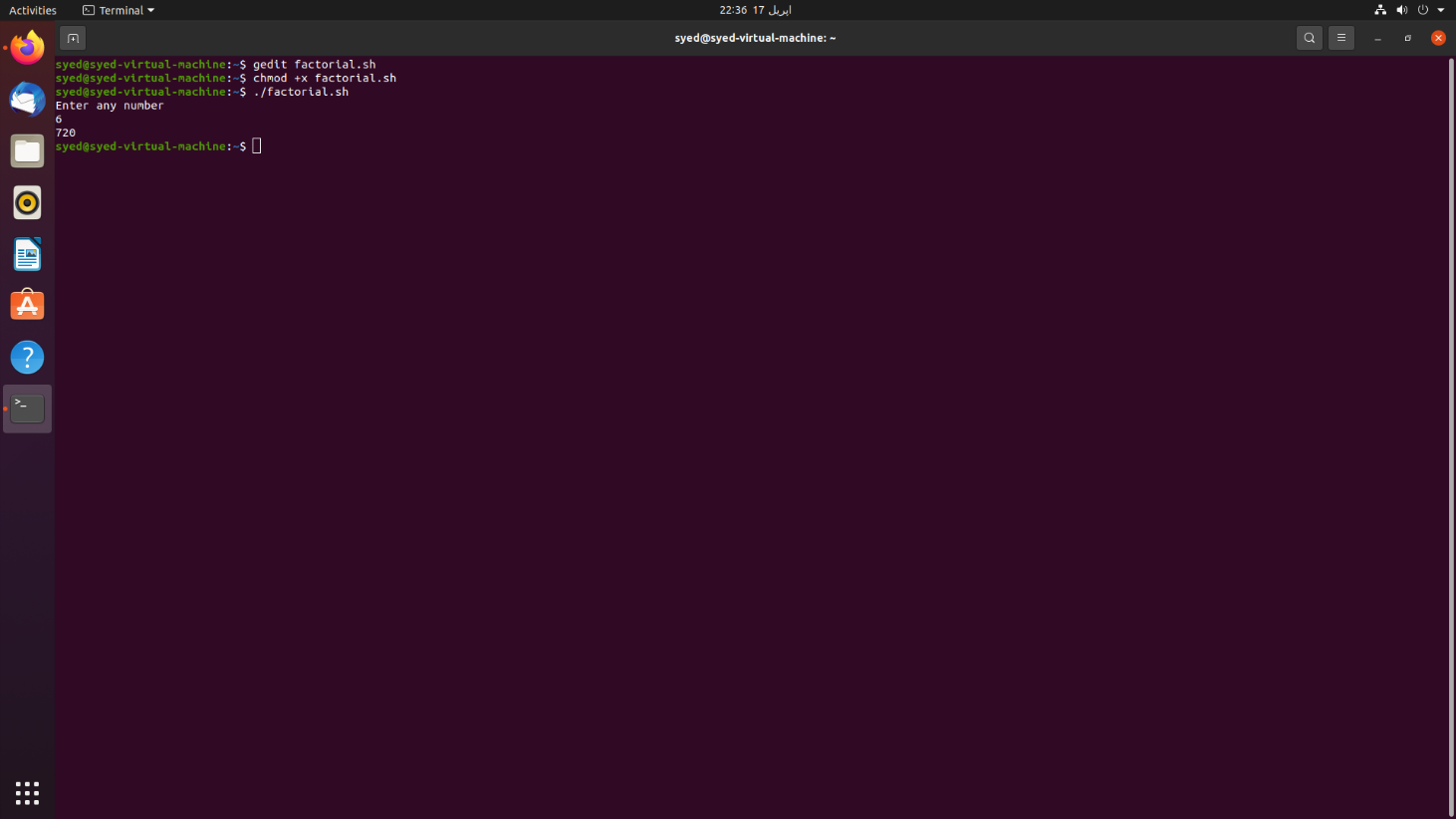
for((i=2;i<=number;i++))

{

f=$((f \* i)) #factorial = factorial \* i

}

echo $f

SCREENSHOT: 

1. Write a shell program to generate Fibonacci series.

CODE:

#!/bin/bash

echo "Enter any number"

read N

m1=0 m2=1

echo "Fibonacci series of this number is : "

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

do

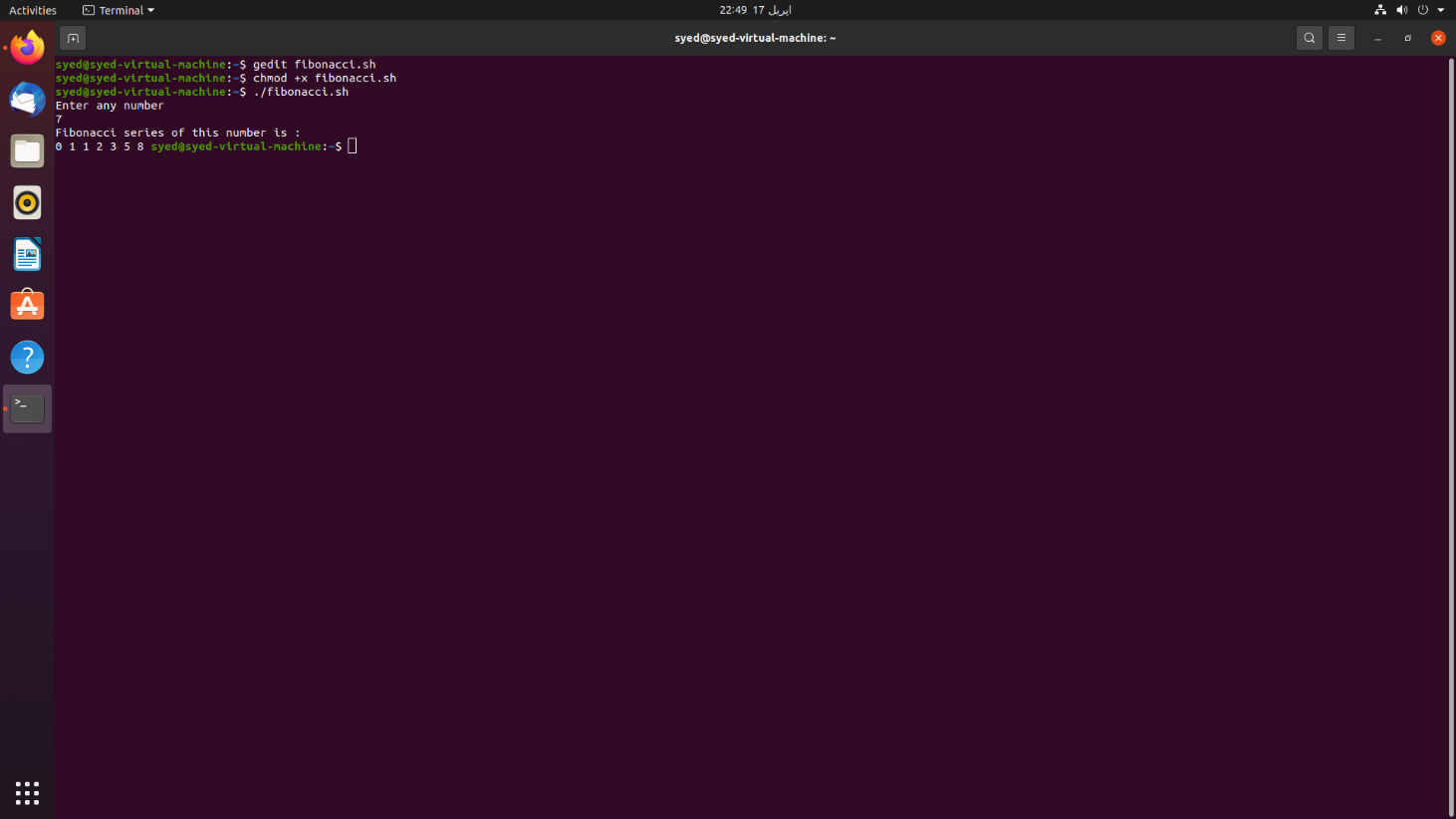
echo -n "$m1 "

fn=$((m1 + m2))

m1=$m2

m2=$fn

done

SCREENSHOT:

**Question 2:**

Think about the use of a three processes with two pipes and implement it.

(You can implement any scenario of your choice).

CODE:

#include <stdio.h>

#include <unistd.h>

int main() {

int pipe1[2],pipe2[2],s1,s2;

char pip1\_m[30]="Hello",pip2\_m[30]="World",read\_m[30];

s1=pipe(pipe1);

if(s1==-1) {

printf("Pipe creation unsuccessfull \n\n");

return 1; }

s2=pipe(pipe2);

if(s2==-1) {

printf("Pipe creation unsuccessfull \n\n");

}

int pid\_t,c1,c2;

c1=fork();

if( c1 != 0 )

{ close(pipe1[0]);

close(pipe2[1]);

printf("parnt process 1,\n message in pipe %s \n",pip1\_m);

write(pipe1[1],pip1\_m,sizeof(pip1\_m));

read(pipe2[0], read\_m,sizeof(read\_m));

printf("parent process 1,\n Rread message in pipe %s \n",read\_m);

}

else{

c2=fork();

if(c2 == 0 ){

close(pipe1[0]);

close(pipe2[1]);

printf("parent process 1, \nmessage in pipe %s \n",pip1\_m); write(pipe1[1],pip1\_m,sizeof(pip1\_m));

read(pipe2[0], read\_m,sizeof(read\_m));

printf("parent process 1, \nread message in pipe %s \n",read\_m);

}

else{

close(pipe1[1]);

close(pipe2[0]);

read(pipe1[0], read\_m, sizeof(read\_m));

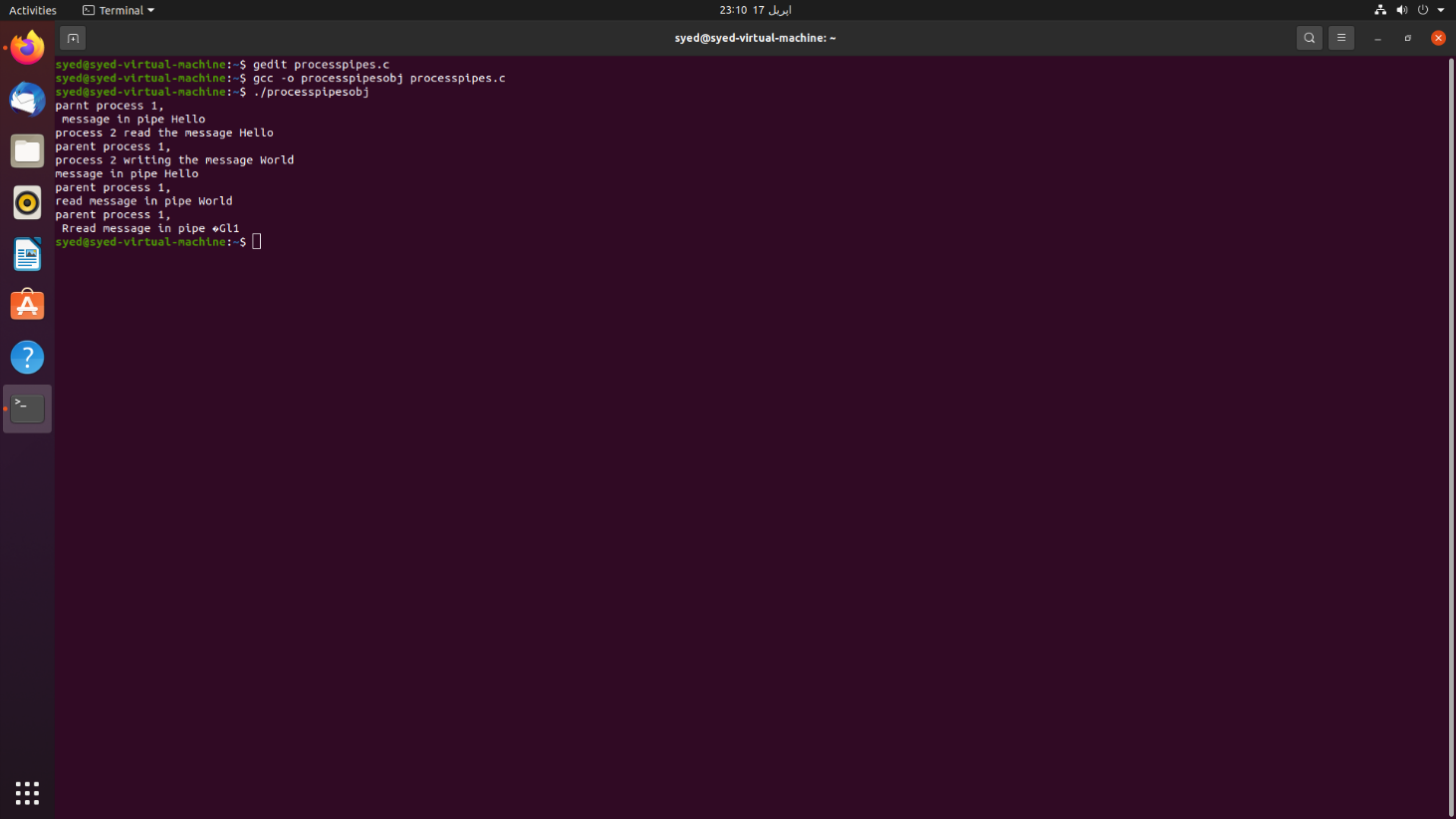
printf("process 2 read the message %s \n" ,read\_m);

printf("process 2 writing the message %s \n",pip2\_m);

write(pipe2[1],pip2\_m,sizeof(pip2\_m));

}}

return 0; }

SCREENSHOT:

**Question 3:**

Consider the following scenario:

There is a ticket booking counter that sells or cancels tickets for a plane seat.

* Initially, there are a total of 10 seats available numbered from 1-10. Only one person can buy or cancel a ticket at a time. A person gets the first seat available from the numbered seats.
* Every person who buys a ticket gets a ticket number and the booked seat number. The first ticket is numbered 1001 and for every successful buy, the number increases by 1.
* If a person cancels a bought ticket, that seat will be made available.
* Implement a program (write two functions **ticket\_buy()** and **ticket\_cancel()** to be called from main()), when there are 20 people (1 to 20) who are standing in a queue in any order to buy a ticket. 10 of these persons from the queue are initially successful in getting a ticket (ticket number 1001 to 1010). Then tickets for 3 seats are cancelled, so three next persons from the queue will get the tickets from the available seats.
* Implement while considering what happens in real life scenario when multiple people want to buy a ticket at the same time and how it is handled.

Hint: Use threads and mutex.

CODE:

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <stdlib.h>

pthread\_mutex\_t mutex1 = PTHREAD\_MUTEX\_INITIALIZER;

int counter = 0;

int seats[10],number,i,j,passeng;

void \*ticket\_buy(){

pthread\_mutex\_lock(&mutex1);

counter++;

printf("Ticket Range: \n");

scanf("%d",&number);

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

printf("Ticket Number: \n");

scanf("%d",&seats[i]);

pthread\_mutex\_unlock(&mutex1);

}

printf("Ticket You Wanna Buy? \n");

scanf("%d",&passeng);

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

{

if(seats[i] == passeng) {

printf("Ticket Booked Successfully: %d \n ",seats[i]);

}

else{

printf("Seats Are Available %d \n ",seats[i]);

}}}

void \*ticket\_cancel(){

pthread\_mutex\_lock(&mutex1);

counter++;

for(i=0; i<seats[i]; i++ ){

printf(" %d \n", seats[i]);

pthread\_mutex\_unlock(&mutex1);

}

printf("Ticket ID You wanna cancel? \n ");

scanf("%d", &seats[i]);

if(seats[i] < 0 || seats[i] > number) {

printf(" Ticket Number %d Cancelled \n", seats[i]);

if(seats[i]==seats[i]){

printf(" Ticket %d is available rn! \n", seats[i]);

}

else{

printf("Seats are not available!");

}}}

int main (){

pthread\_t t1,t2;

int R1,R2,i,select;

for(i=0;i<3;i++){

printf("Pls Enter 1 For Booking \nPls Enter 2 For Cancellation:\n ");

scanf("%d",&select);

if( select == 1){

printf("Buy Ticket: \n");

R1=pthread\_create(&t1,NULL,ticket\_buy,NULL);

pthread\_join(t1,NULL); }

else if(select == 2){

printf("Cancelling Ticket ID: \n");

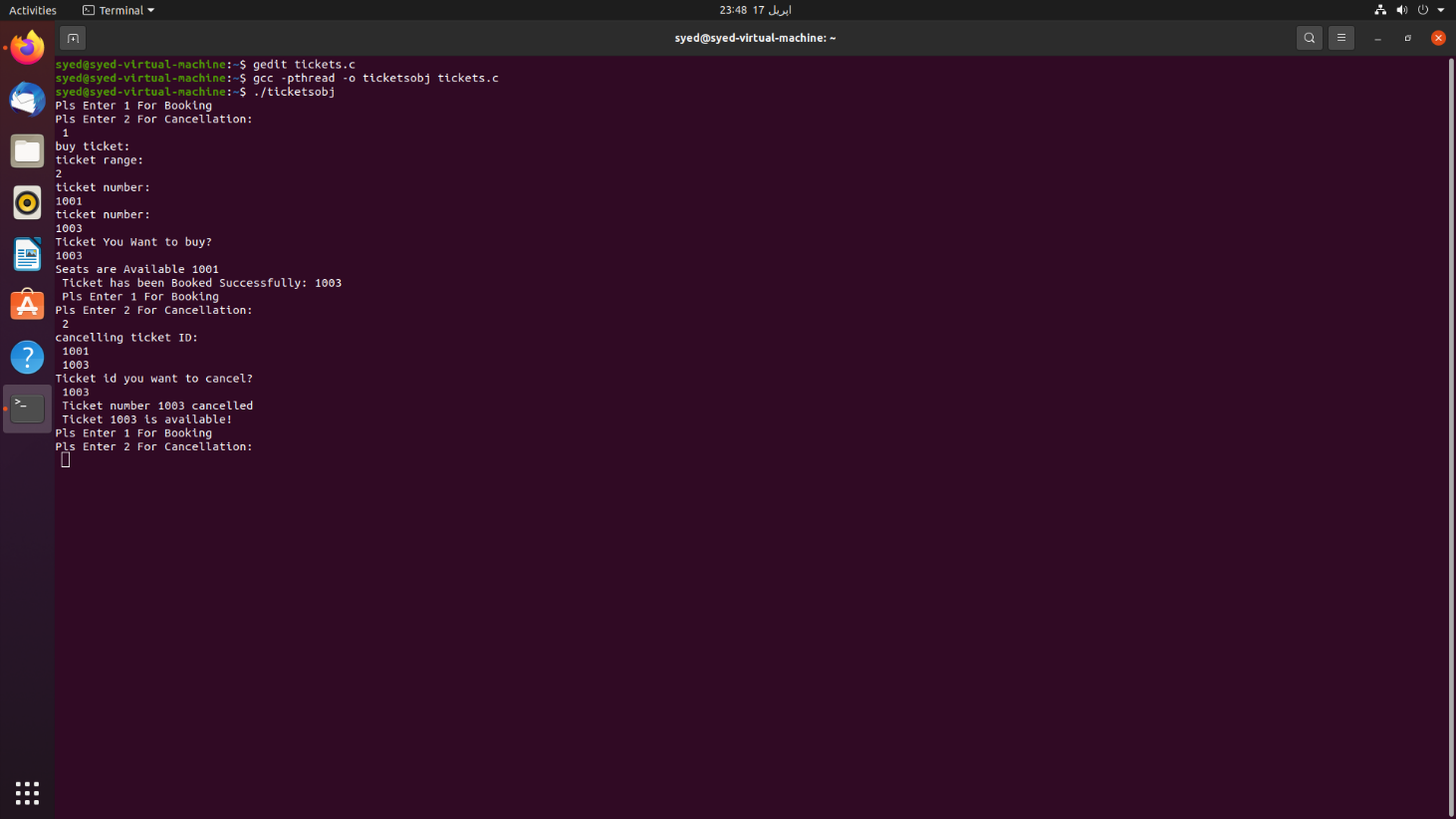
R2=pthread\_create(&t2,NULL,ticket\_cancel,NULL);

pthread\_join(t2,NULL);

}}

return 0;

}

SCREENSHOT: