**Operating System Lab**

**Lab Exercise #3**



**Session: 2021**

**Submitted by:**

**Wali Muhammad 2021-SE-39**

**Submitted to:**

**Mr. Waqas Ali**

Department of Computer Science, New Campus

**University of Engineering and Technology Lahore, Pakistan**

**Task 1**

**Description**

Create a child process using the fork system call then print the process id and parent process id of all running processes.

**Solution**

**Code**

#include<iostream>

#include<unistd.h>

using namespace std;

int main()

{

cout<<"Current Process ID: "<<getpid()<<endl;

cout<<"Current Process Parent's ID: "<<getppid()<<endl;

pid\_t pid = fork();

if(pid==0)

{

cout<<"Child Process ID: "<<getpid()<<endl;

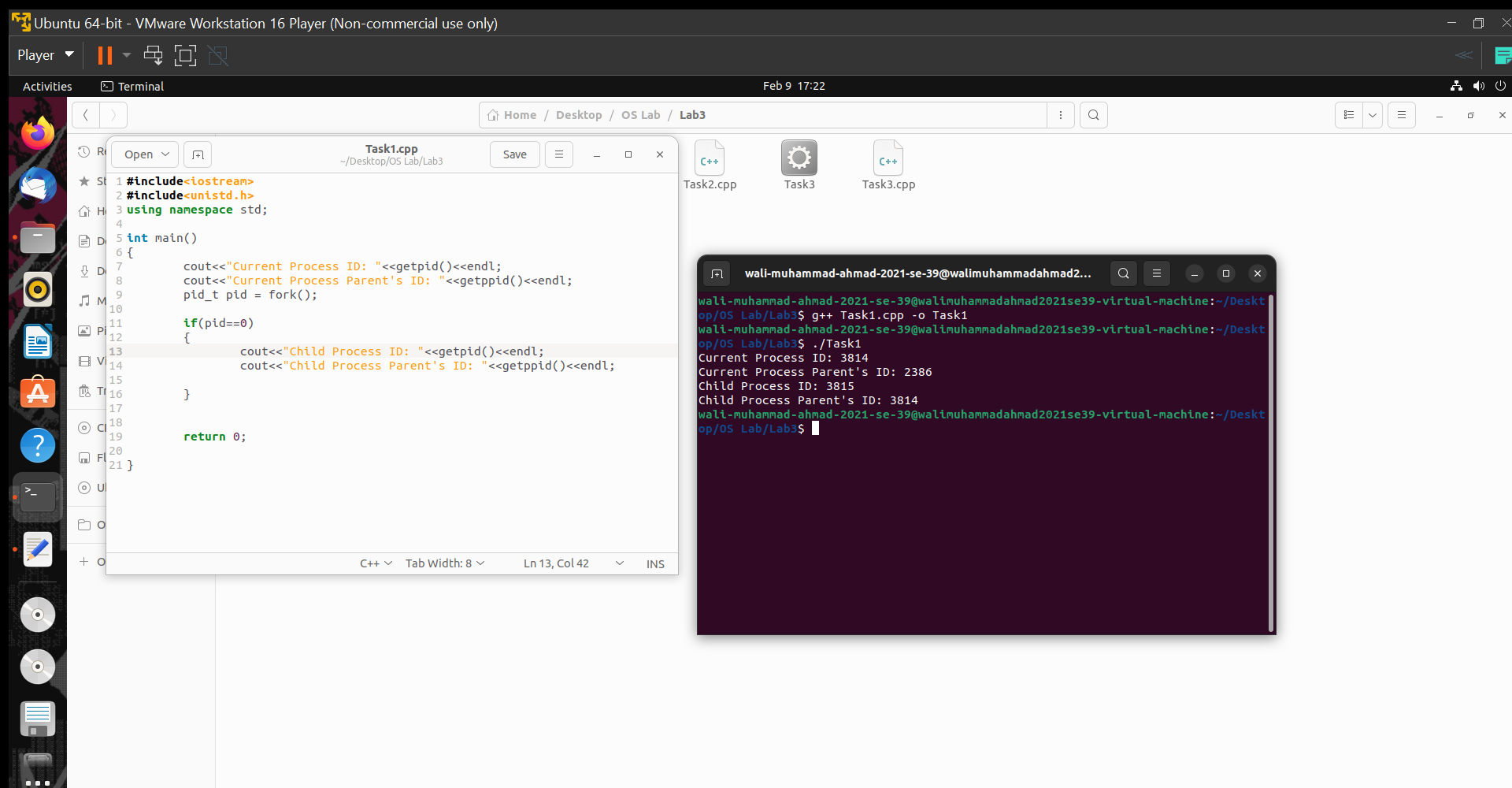
cout<<"Child Process Parent's ID: "<<getppid()<<endl;

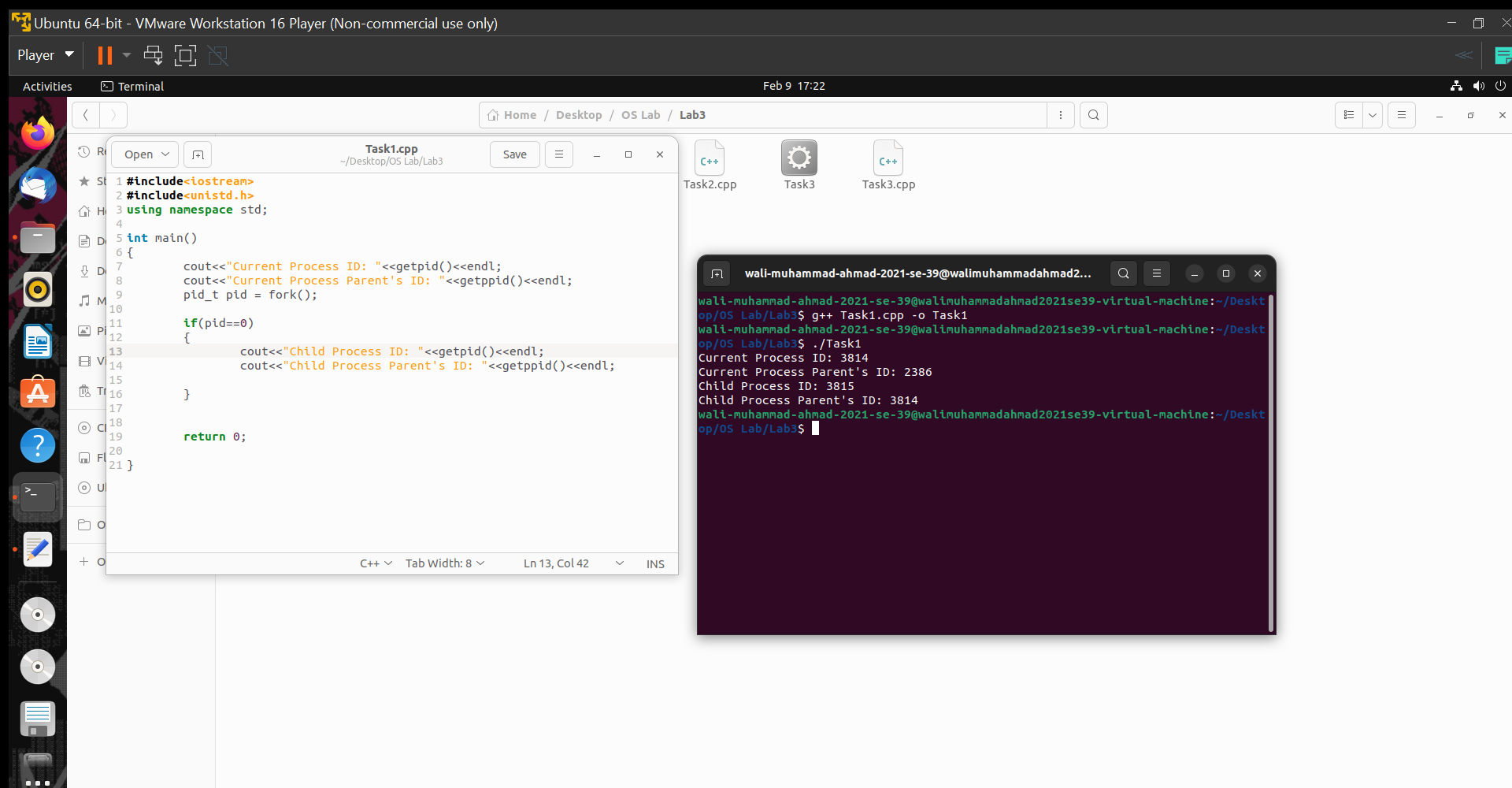
}

return 0;

}

**Code Execution Screenshot**

****

****

**Task 2**

**Description**

Execute the following loop in your program

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

{

fork();

}

cout << "Hello from the process " << getpid() << endl;

predict the output of the program,

is it same as your predicted output?

**Solution**

**Code**

#include<iostream>

#include<unistd.h>

using namespace std;

int main()

{

pid\_t pid;

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

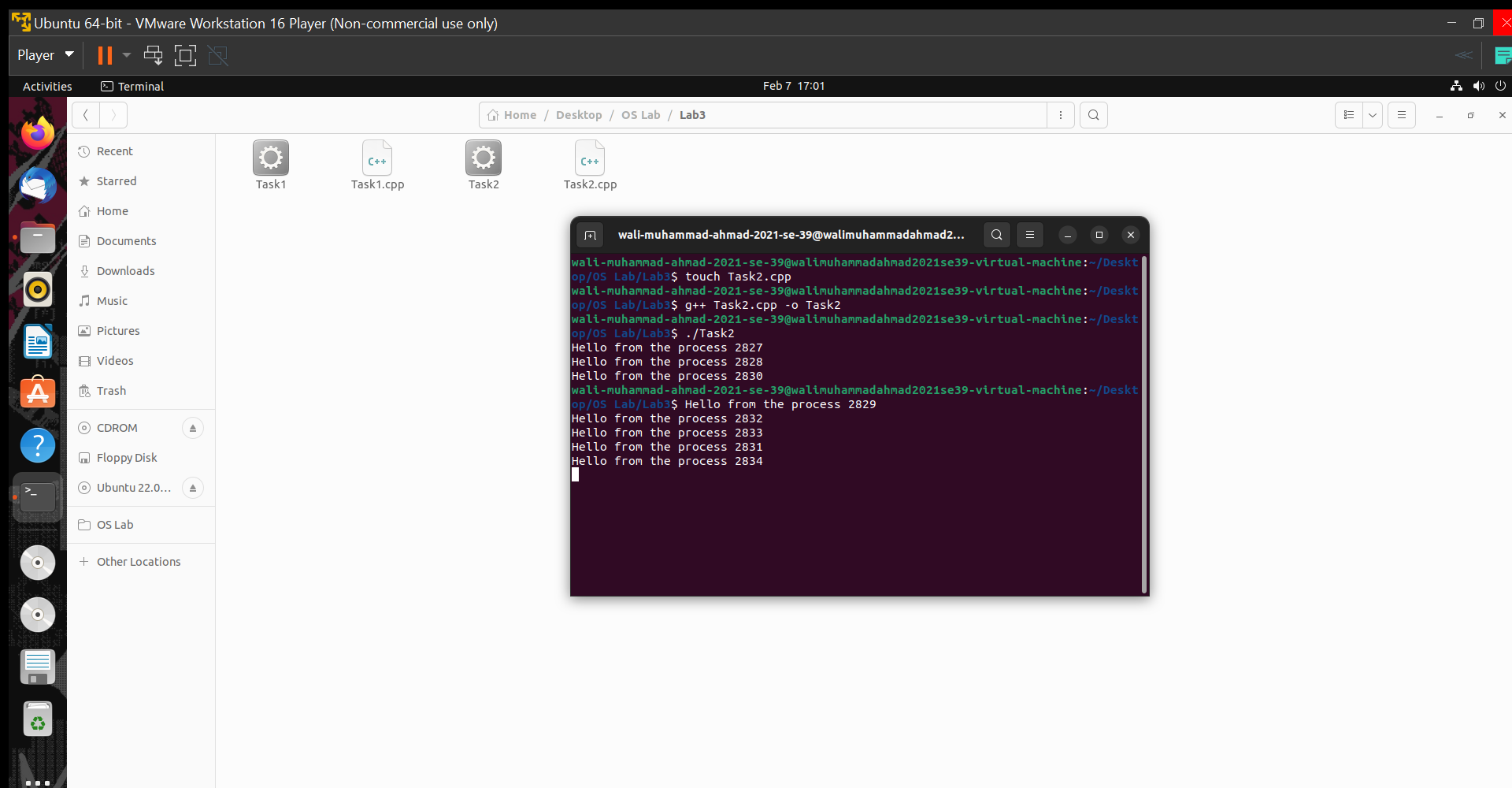
{fork();}

cout<<"Hello from the process "<<getpid()<<endl;

return 0;

}

**Code Execution Screenshot**



**Task 3**

**Description**

Write a program that launches four processes using fork system call

* Process 0 display the number between 1 and 25
* Process 1 displays the numbers between 26 and 50
* Process 2 displays the numbers between 51 and 75
* Process 3 displays the numbers between 76 and 100

**Solution**

**Code**

#include<iostream>

#include<unistd.h>

using namespace std;

int main()

{

int num = 1;

int count = 25; //variable for loop iteration

int rank = 1; //rank of procesess

int process = 0; //Number of process

for (int i = 1; i <= 2; ++i)

{

if (fork() == 0)

{

rank = rank + i;

break;

}

}

for (int i = 0; i <= rank; i++)

{

cout<<"Process "<< process <<endl;

for (; num < count; num++)

cout << num << " ";

cout << endl;

count = count + 25; //for increment of next 25

process++;

}

return 0;

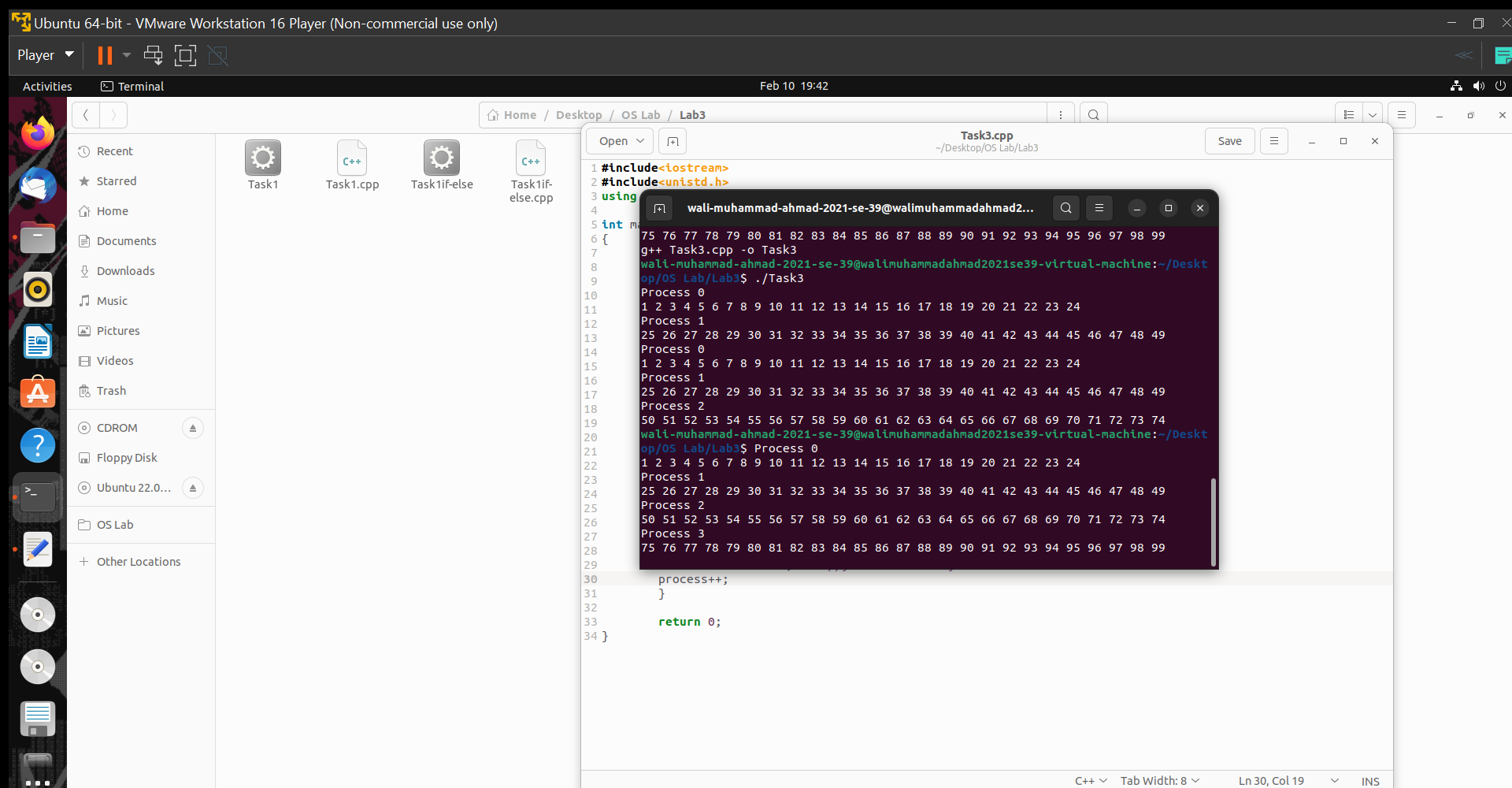
}

**Code Explanation**

The loop will execute 2 times and fork () will be called 2 times so there will be total of 4 processes. At first iteration, the fork () creates 1 child process and after second iteration the fork () create 2 child processes. So, there will be 4 total processes.

Parent process is process = 0, 1st fork () creates 1 new Child process that is process = 1, 2nd fork () creates 2 new child processes that are process = 3 and process = 4 respectively. Every process prints 24 integers on terminal.

**Code Execution Screenshot**



**Task 4**

**Description**

Write a program that launches four processes using fork system call, then all the processes

counts that how many prime numbers exists between 2 and 100,001.

Now process 0 should find the count between --- 2 to 25,001

process 1 should find the count between --- 25,002 to 50,001

process 2 should find the count between --- 50,002 to 75,001

process 3 should find the count between --- 75,002 to 100,001

Below there a function is given which finds out whether a number if prime or not

bool isPrime(int num)

{

if (num == 1)

{

return true;

}

else

{

double result;

int divisor = num - 1;

while (num != -1)

{

result = num % divisor;

if (result == 0)

{

num = -1;

}

else

{

divisor = divisor - 1;

}

}

if (divisor == 1)

{

return true;

}

else

{

return false;

}

}

**Solution**

**Code**

#include<iostream>

#include<unistd.h>

using namespace std;

int isPrime(int num)

{

int prime = num;

bool flag = false;

if (num == 1)

{

flag = true;

}

else

{

int result;

int divisor = num - 1;

while (num != -1)

{

result = num % divisor;

if (result == 0)

{

num = -1;

}

else

{

divisor = divisor - 1;

}

}

if (divisor == 1)

{

flag = true;

}

else

{

flag = false;

}

if (flag == true)

{

return prime;

}

else

return 0;

}

return 0;

}

int main()

{

int num = 1;

int start = 2;

int increment = 25000; //variable for loop iteration

int rank = 1; //rank of procesess

int counter = 0; //counter for total prime number

for (int i = 1; i <= 2; ++i)

{

if (fork() == 0)

{

rank = rank + i;

break;

}

}

for (int i = 0; i <= rank; i++)

{

int prime = 0; //counter for prime number in a process

for (; num < increment; num++)

{

if (isPrime(num) != 0)

{

counter++;

prime++;

}

}

cout << "Prime numerbers found between " << start << " and " << increment << " are " << prime << endl;

increment = increment + 25000; //for increment of next 25000

start = start + num;

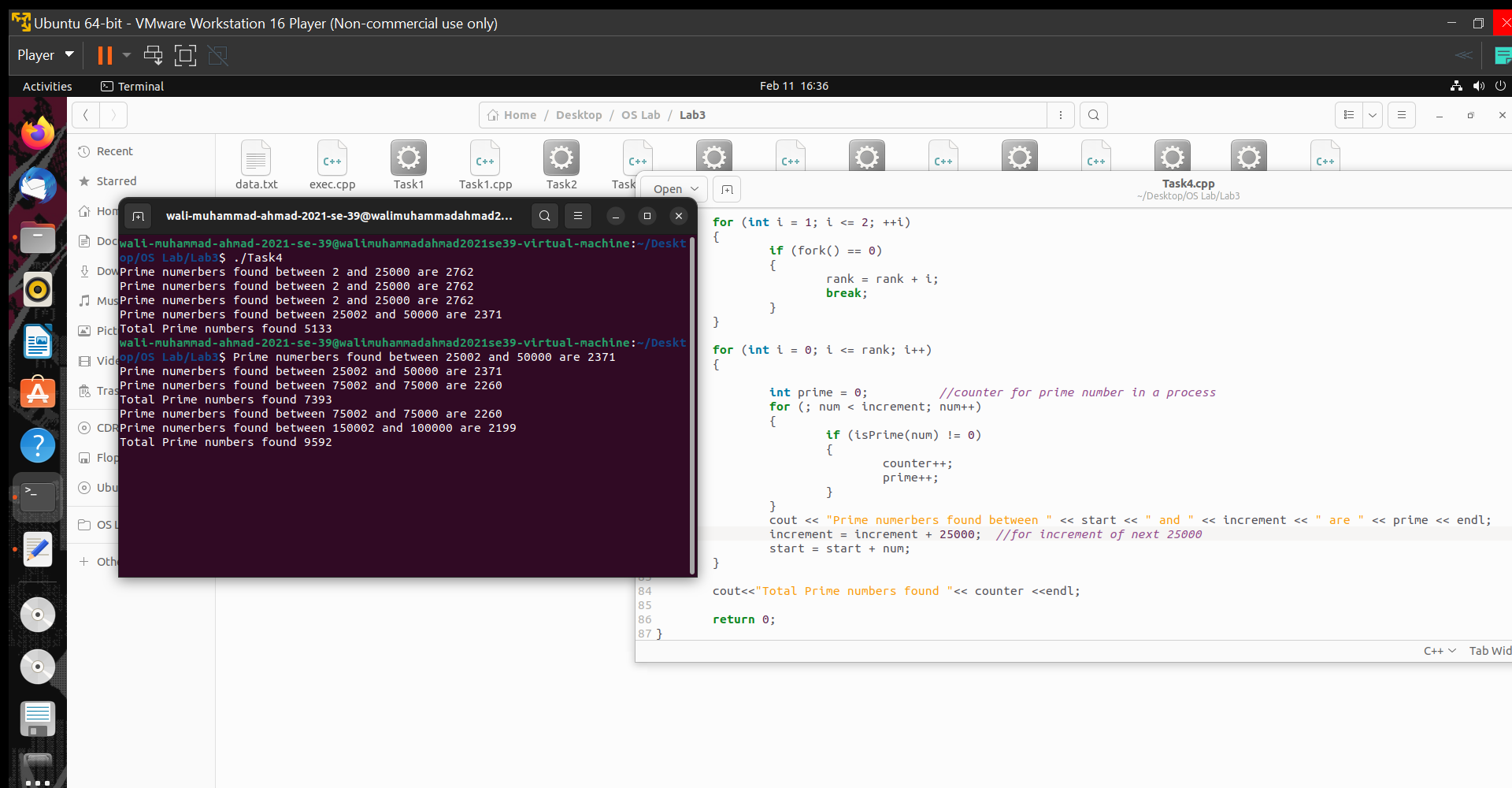
}

cout<<"Total Prime numbers found "<< counter <<endl;

return 0;

}

**Code Execution Screenshot**

****

According to internet there are 9593 prime numbers between 1 to 100001 so the answer is correct that the prime number between 2 to 100001 is 9592 (Excluding 1).

**Task 5**

**Description**

Write a C/C++ Program to create a great grandchild of a process i.e.,

Parent-> Child -> Child -> Child,

Do incorporate checks to ensure that no extra siblings

are created.

**Solution**

**Code**

#include <iostream>

#include <unistd.h>

using namespace std;

int main()

{

cout<<"Parent PID "<<getpid()<<endl;

pid\_t pid1, pid2, pid3;

pid1 = fork();

if (pid1 < 0)

{

cout << "Process creating unsuccessful" << endl;

}

else if (pid1 == 0)

{

cout<<"Child PID "<<getpid()<<endl;

cout<<"Child parent's PID "<<getppid()<<endl;

pid2 = fork();

if (pid2 < 0)

{

cout << "Process creating unsuccessful" << endl;

}

else if (pid2 == 0)

{

cout<<"Grand Child PID "<<getpid()<<endl;

cout<<"Grand child parent's PID "<<getppid()<<endl;

pid3 = fork();

if (pid3 == 0)

{

cout << "Great grandchild process." << endl;

cout<<"Great Grandchild PID "<<getpid()<<endl;

}

else if (pid3 < 0)

{

cout << "Process creating unsuccessful" << endl;

}

}

}

return 0;

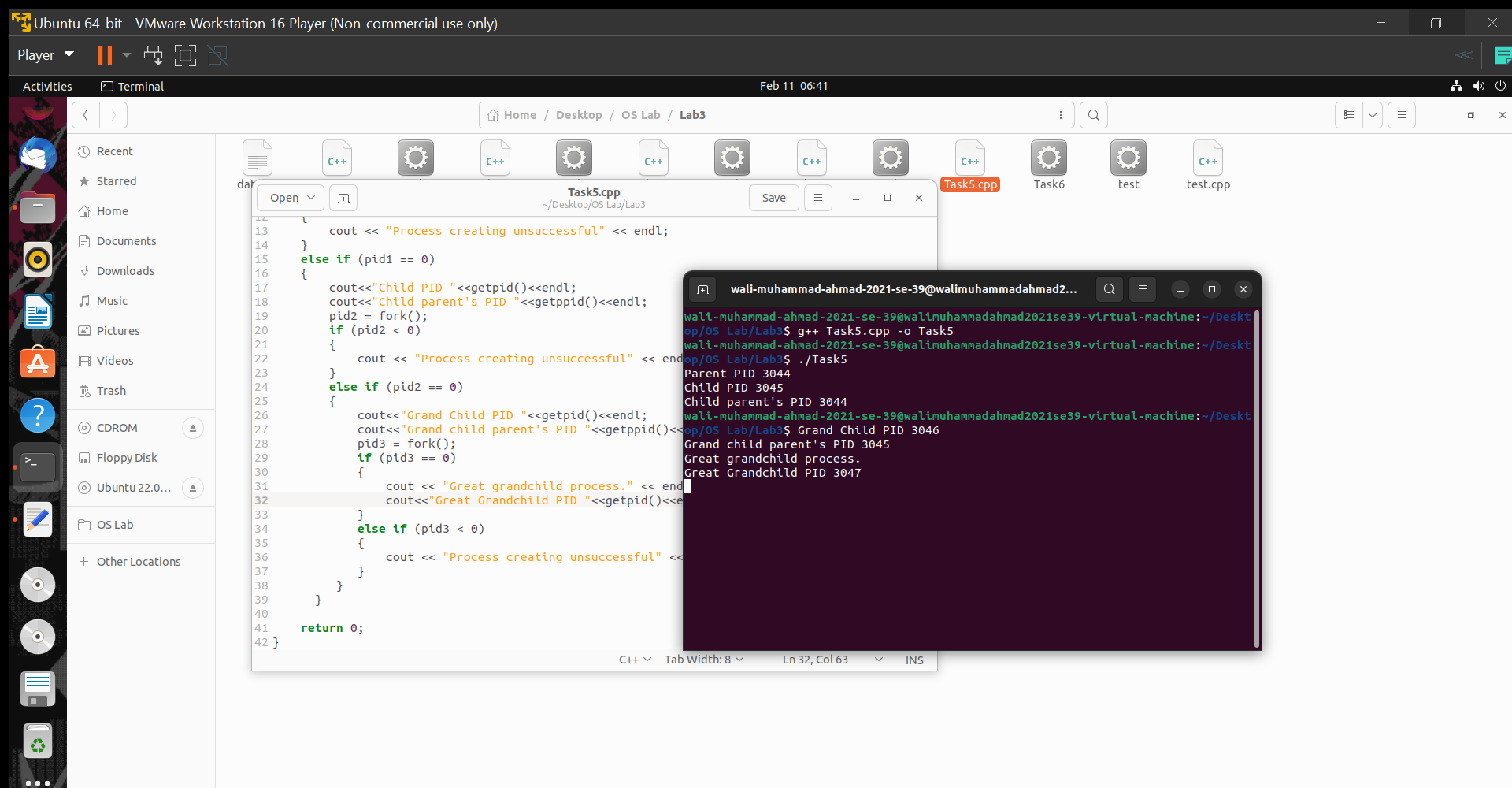
}

**Code Explanation**

Nested if-if else statement is used to create Great grand child which has no siblings. Every process ID is shown with the process so it is easy to demonstrate the Child process chain

Parent-> Child -> Child -> Child.

**Code Execution Screenshot**

****

**Task 6**

**Description**

To load an executable program in a child processes exec system call.

**Algorithm**

1. Firstly create a text file named data.txt in which write “My name is \_\_\_\_\_\_\_\_\_ (YOUR

NAME) and my roll number is \_\_\_\_\_\_\_\_ (YOUR ROLL NUMBER) “.

2. Create a program named “test.c”. In this program read this text file.

3. Now create a new program named “exec.c”. In this program create a child process and in that

child load the program “test.c”.

4. If return value is -1 then

a. Print "Process creation unsuccessful"

b. Terminate using exit system call.

5. Stop.

**Solution**

**Code**

**test.cpp content**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

int main()

{

fstream file;

string data;

string n;

file.open("data.txt", ios :: in | ios :: out);

while(getline(file,data))

{

cout<<data<<endl;

}

file.close();

return 0;

}

**exec.cpp content**

#include <iostream>

#include <unistd.h>

using namespace std;

int main()

{

pid\_t pid = fork();

if (pid == 0)

{

execlp("./test", "./data.txt", NULL);

cout<<"Child Process ended"<<endl; //wont execute

}

else if (pid < 0)

{

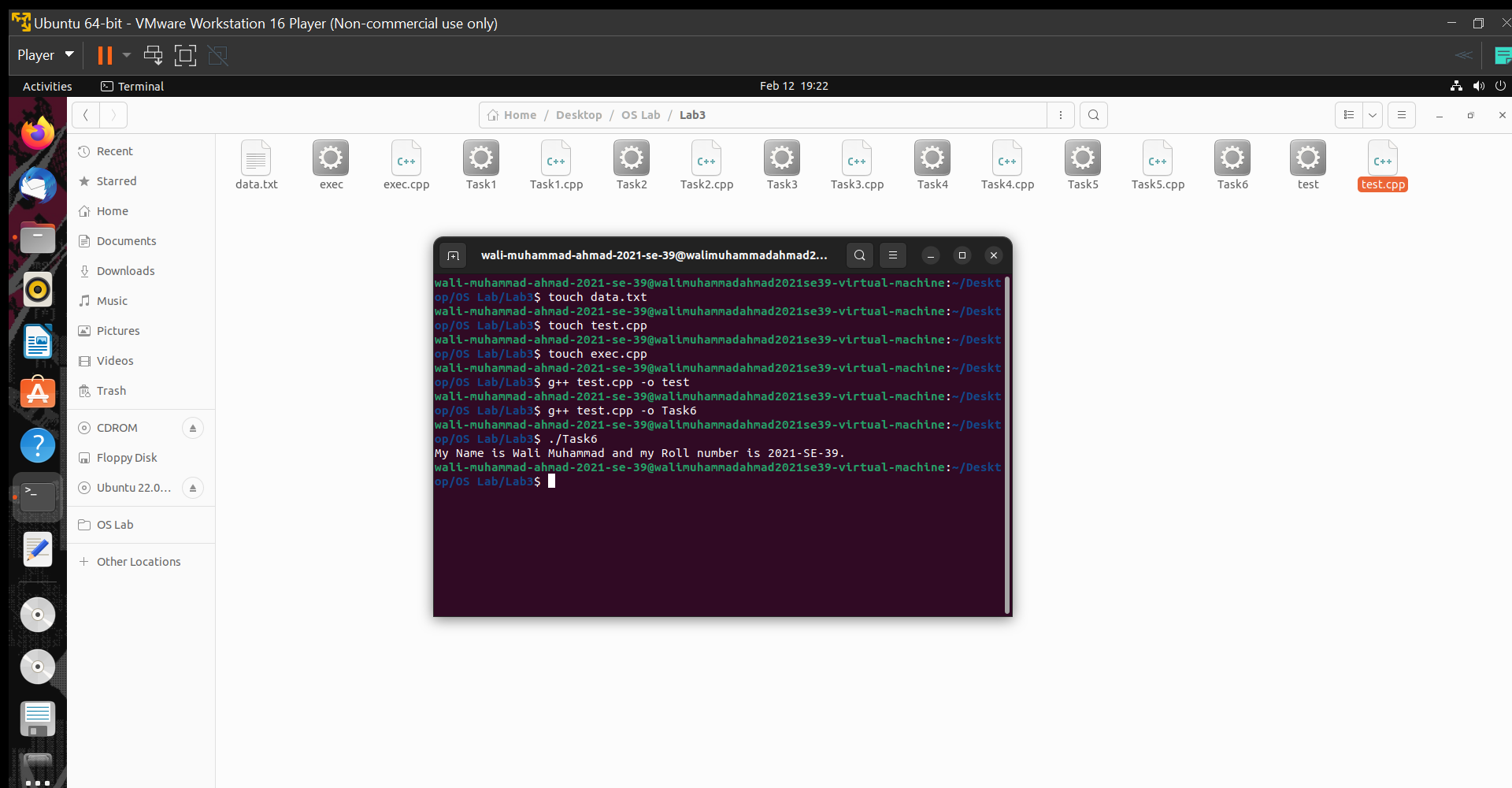
cout << "Process creation unsuccessful." << endl;

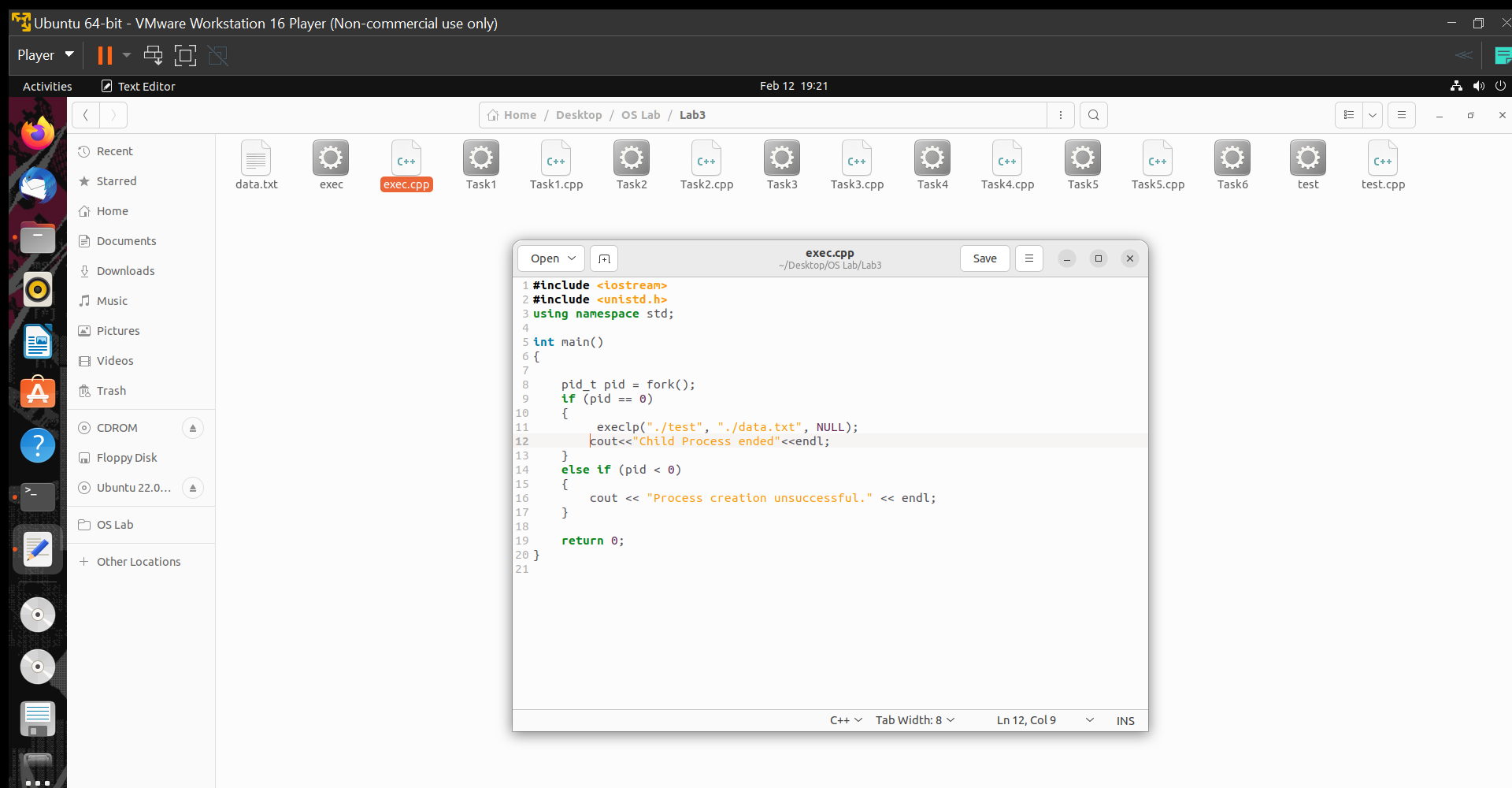
}

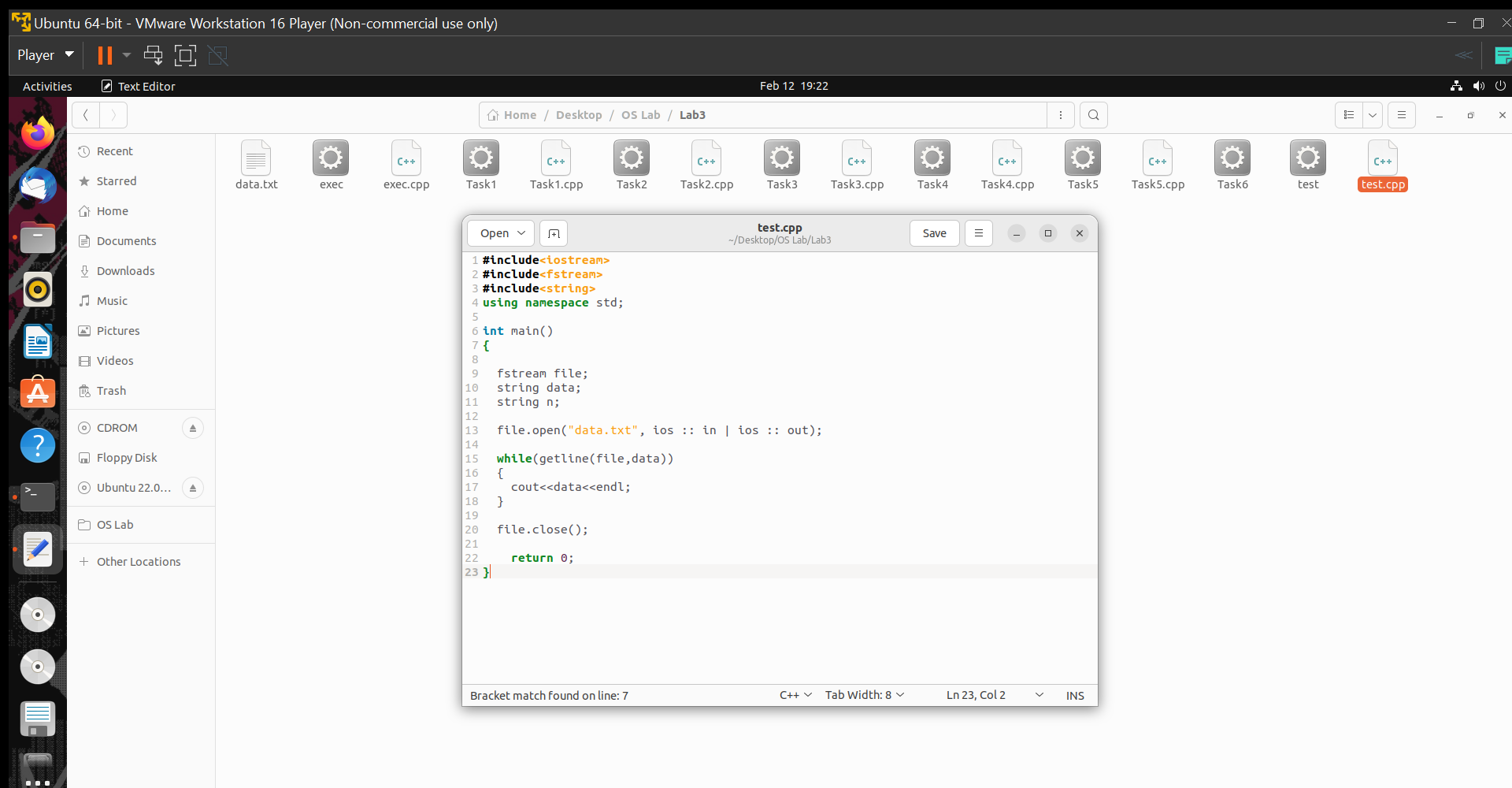
return 0;

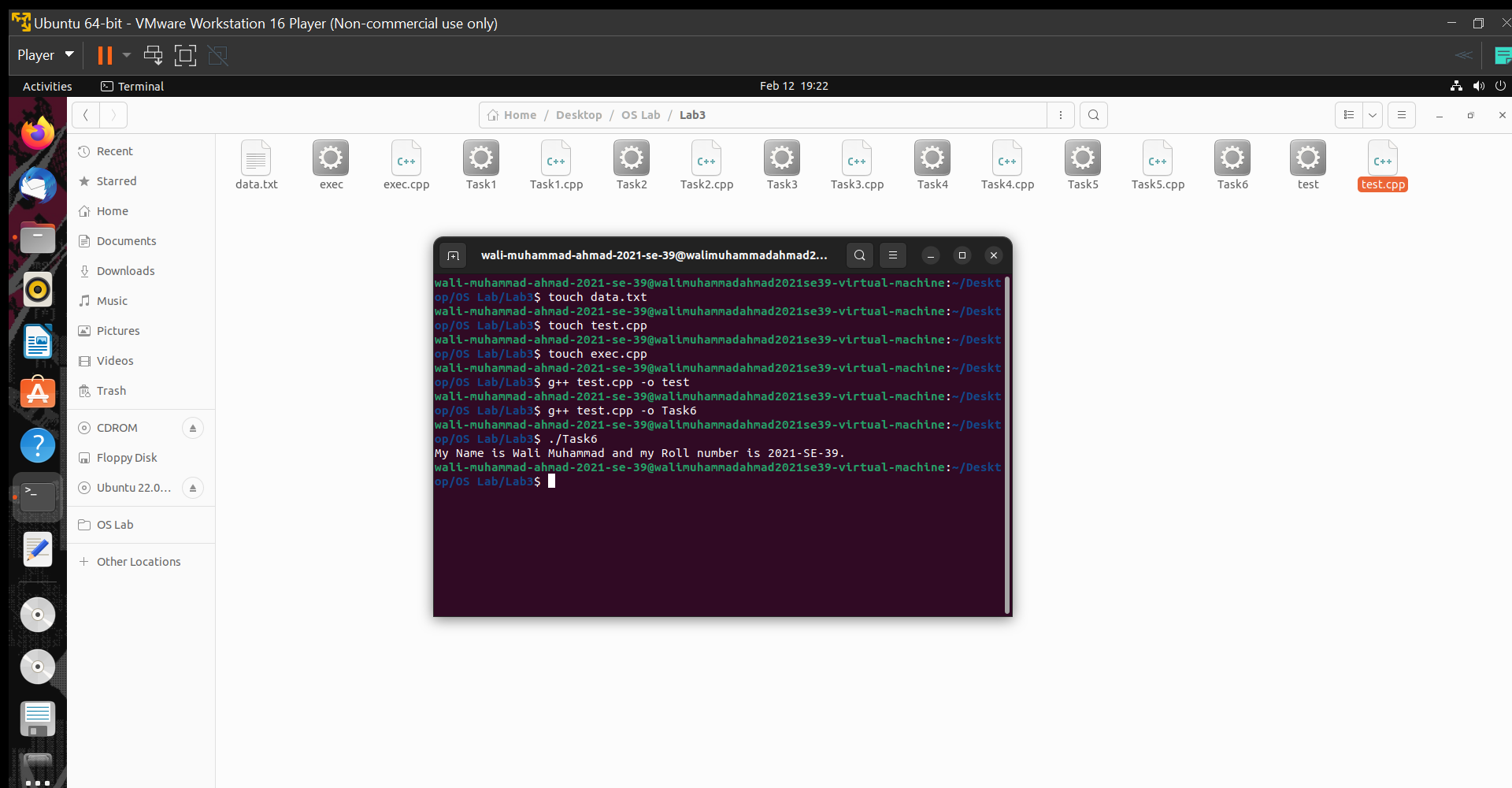
}

**Code Execution Screenshot**

****

****

****

****

**THE END**