Lab Tasks

By

Tooba Baqai
46489

Submitted to: Ma’am Kausar

Subject: Operating System

Date:10/8/2024

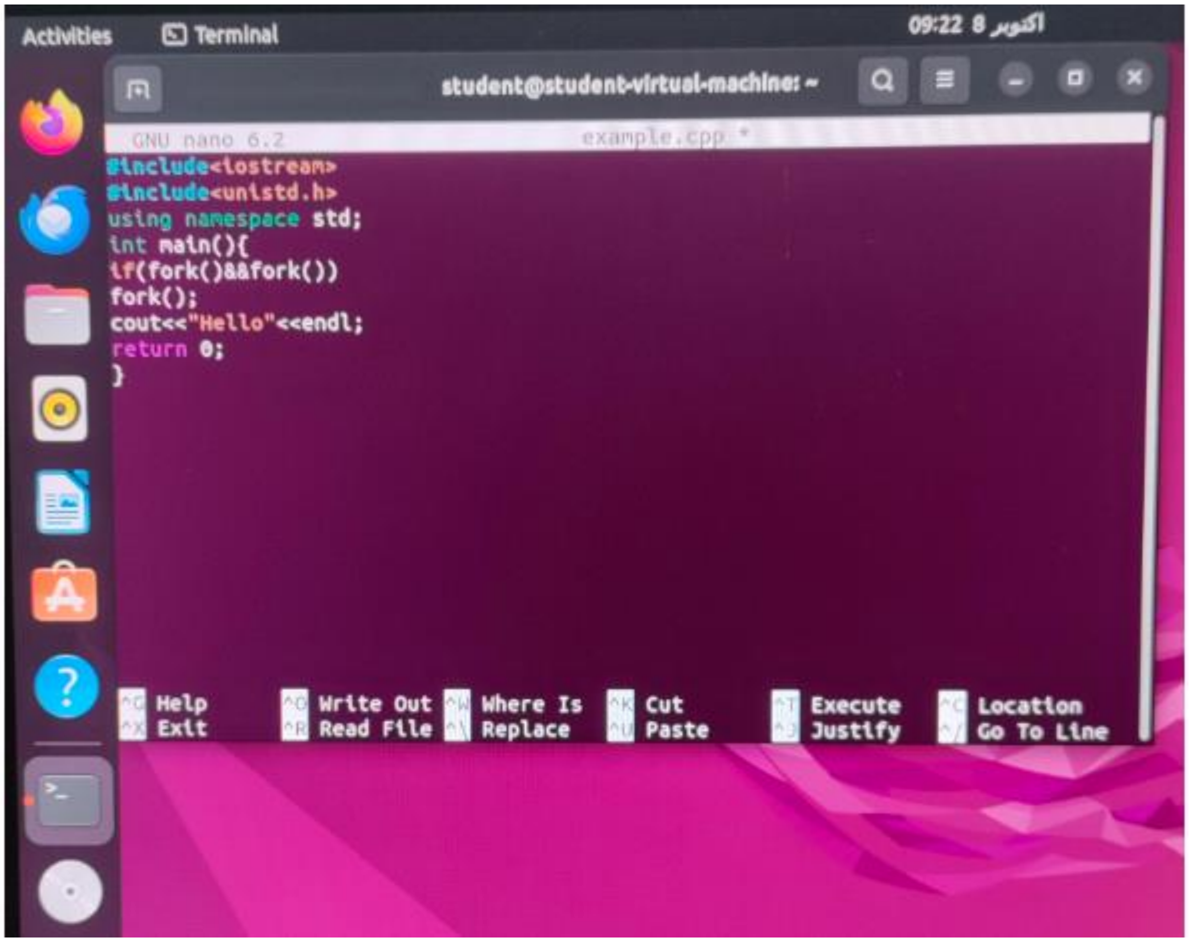
BSCS SEMESTER – 5

RIPHAH INTERNATIONAL UNIVERSITY

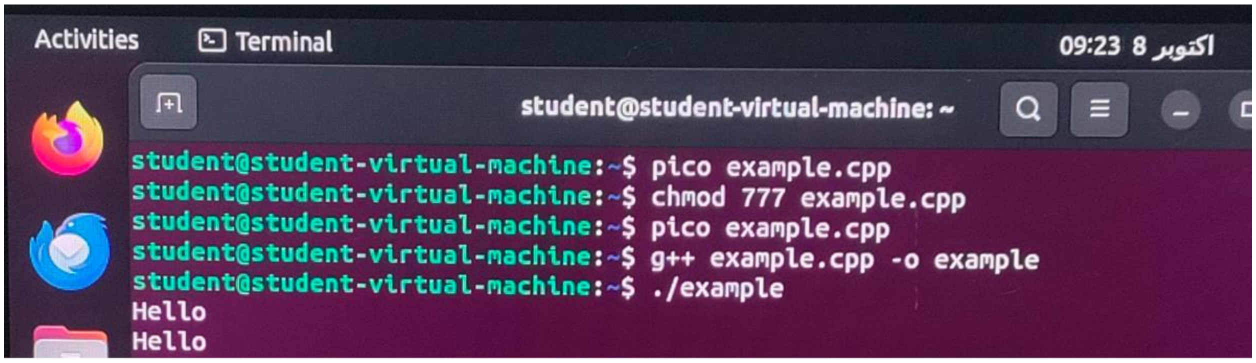
ISLAMABAD, PAKISTAN

Task

1. **Write a C/C++ program that uses the fork() function and the logical AND (&&) operator.**



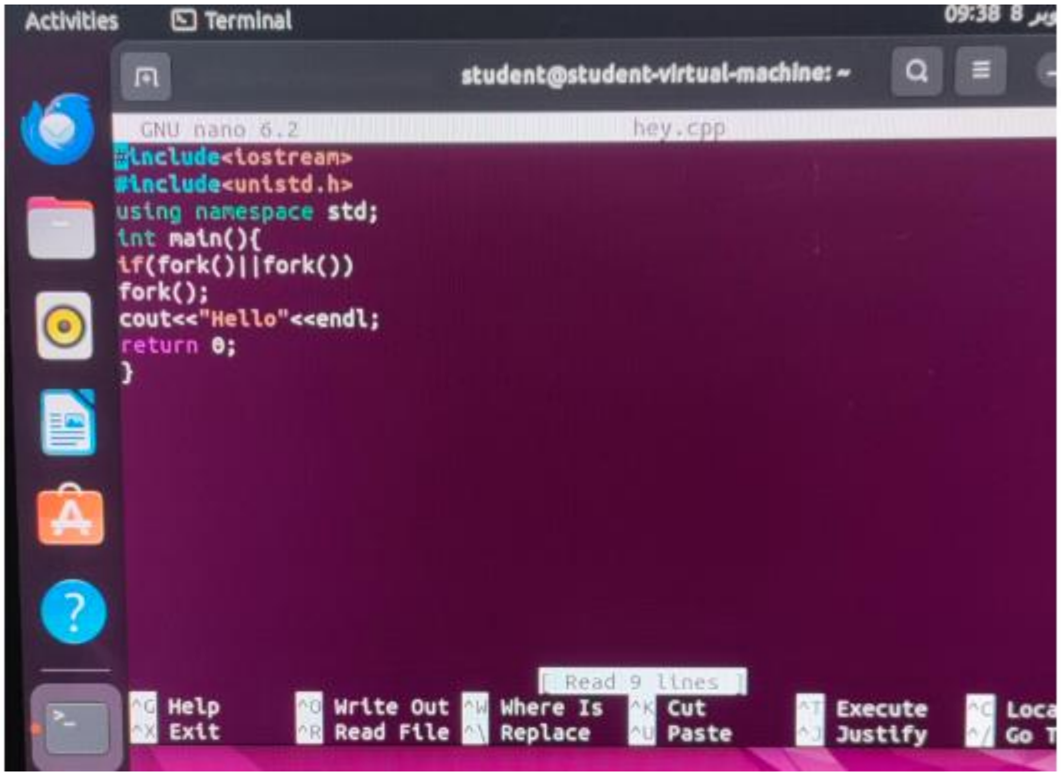
```
#include<iostream>
#include<unistd.h>
using namespace std;
int main(){
if(fork()&&fork())
fork();
cout<<"Hello"<<endl;
return 0;
}
```



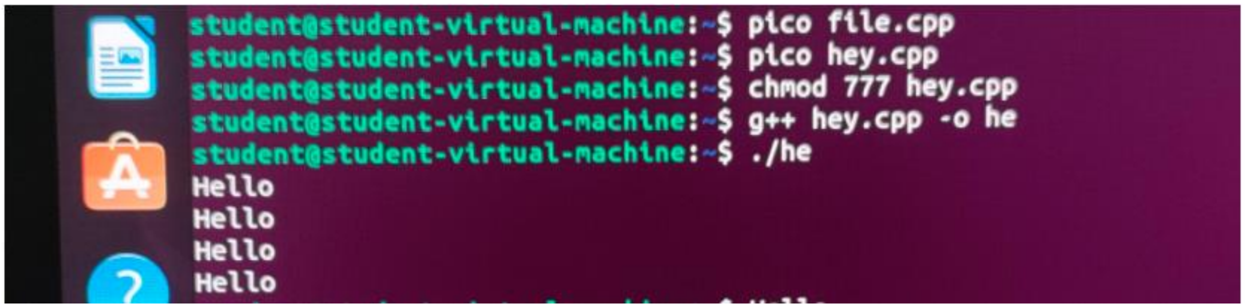
```
student@student-virtual-machine:~$ pico example.cpp
student@student-virtual-machine:~$ chmod 777 example.cpp
student@student-virtual-machine:~$ pico example.cpp
student@student-virtual-machine:~$ g++ example.cpp -o example
student@student-virtual-machine:~$ ./example
Hello
Hello
```

In the case of the AND operator, if both conditions are true, the program proceeds accordingly. In this scenario, a fork creates a child process. When the first fork is executed, it creates a child process (with PID 0) and the parent process (with a non-zero PID, typically 1). After this, the second fork is invoked by the parent (PID 1), while the child (PID 0) terminates. The parent process (PID 1) is then split into a new child (PID 0) and a parent (PID 1). The last fork is only executed by the parent (since the condition 1 + 1 = true), and the child process is again terminated. This results in the output of "hello" being printed twice..

2. **Write a C/C++ program that uses the fork() function and the logical OR (II) operator.**

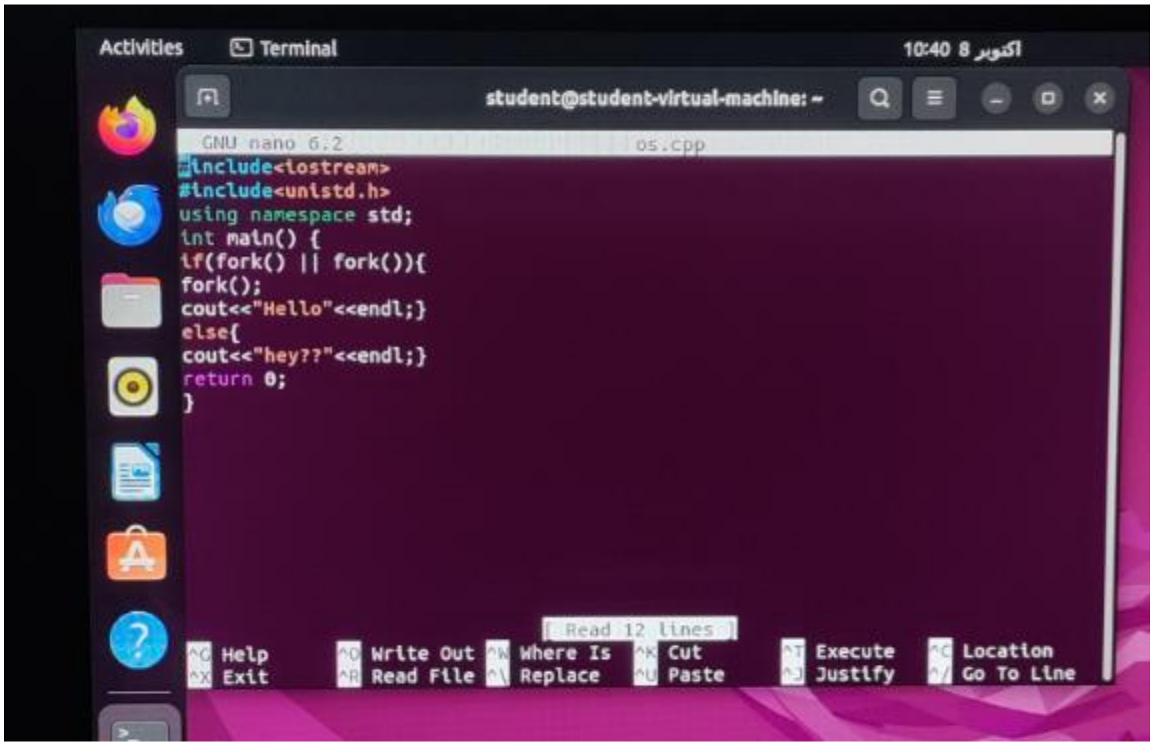


```
#include<iostream>
#include<unistd.h>
using namespace std;
int main(){
if(fork()||fork())
fork();
cout<<"Hello"<<endl;
return 0;
}
```



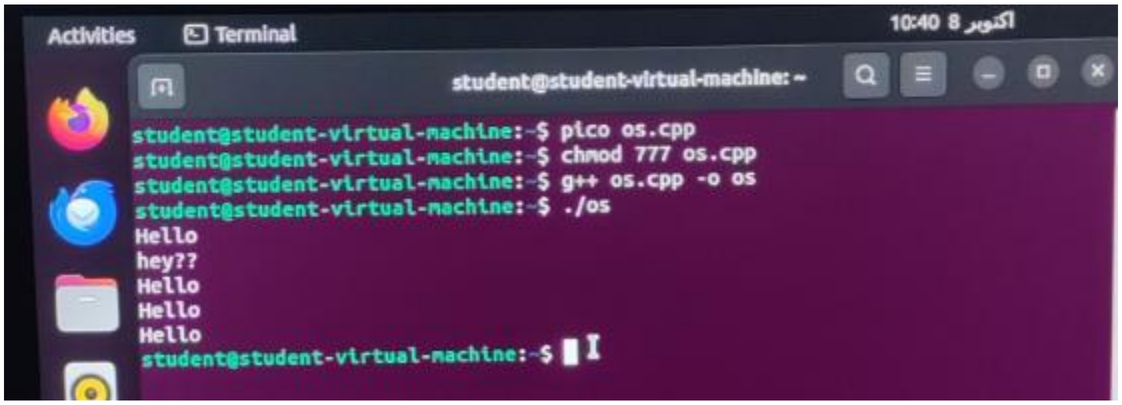
```
student@student-virtual-machine:~$ pico file.cpp
student@student-virtual-machine:~$ pico hey.cpp
student@student-virtual-machine:~$ chmod 777 hey.cpp
student@student-virtual-machine:~$ g++ hey.cpp -o he
student@student-virtual-machine:~$ ./he
Hello
Hello
Hello
Hello
```

3. In the case of the OR operator, if any one condition is true, the program proceeds. In this example, the fork function creates child processes. When the first fork is executed, it generates a child process (PID 0) and a parent process (PID 1). The second fork is called for both the parent (PID 1) and the child (PID 1). The parent (PID 1) is then split into a new child (PID 0) and a parent (PID 1), while the child (PID 1) is also split into two new child processes (both with PID 0). Since one condition in child (PID 1) is true (0 + 1 = true), it undergoes further splitting into two more child processes (both with PID 0). This is the final call to fork. As a result, "hello" is printed four times.
4. **Write a C++ program that uses fork() to create a child process. Use an if-else statement.**



```
  GNU nano 6.2                          os.cpp
#include<iostream>
#include<unistd.h>
using namespace std;
int main() {
if(fork() || fork()){
fork();
cout<<"Hello"<<endl;}
else{
cout<<"hey??"<<endl;}
return 0;
}

                          [ Read 12 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```



```
student@student-virtual-machine:~$ pico os.cpp
student@student-virtual-machine:~$ chmod 777 os.cpp
student@student-virtual-machine:~$ g++ os.cpp -o os
student@student-virtual-machine:~$ ./os
Hello
hey??
Hello
Hello
Hello
student@student-virtual-machine:~$
```

In this program, we use the AND operator with an if-else statement. If the fork fails (i.e., the condition in the if statement is false), the else block will execute. With the AND operator, if one of the conditions is false, the entire condition evaluates as false, preventing further execution of the following statements. Therefore, if the fork fails, the else case will run.