RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD

Lab#8

Bachelors of Computer Science – 5th Semester

Subject: Operating System

Submitted to: Ms. Kausar Nasreen Khattak

Submitted by: Tabinda Hassan

SAP-46374

Date of Submission: 08-Oct-2024

Q1: Write a C/C++ program that uses the fork() function and the logical AND (&&) operator.

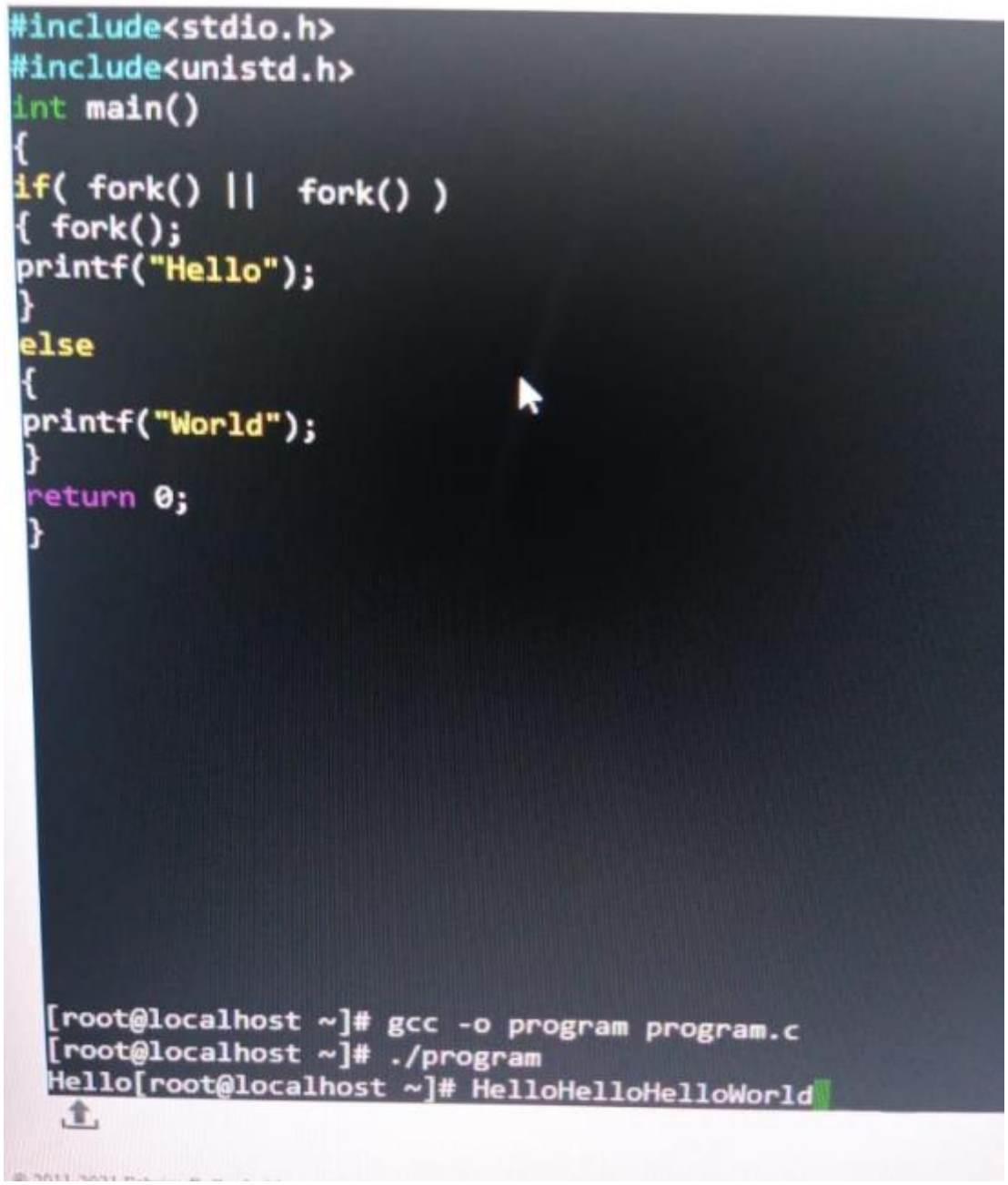
```
 GNU nano 5.3                          program.c
#include<stdio.h>
#include<unistd.h>

int main()
{
   if( fork() && fork())
{
   fork();
   printf("Hello");
}
  else
{
   printf("Bye");
}
  return 0;
}
```

```
[root@localhost ~]# gcc -o program program.c
[root@localhost ~]# ./program
[root@localhost ~]# ./program
Hello[root@localhost ~]# HelloByeBye
```

Explanation:

In this program, the `fork()` function is used to create a child process. The logical AND (`&&`) operator is used in a condition to execute a block of code only if both conditions are true. The `&&` operator ensures that the second condition is evaluated only if the first one is true. This is helpful in situations where we want to perform some task in both parent and child processes based on certain conditions.

Q2: Write a C/C++ program that uses the fork() function and the logical OR (II) operator.



Explanation:

In this program, the `fork()` function creates a child process, and the logical OR (`||`) operator is used in a condition. The `||` operator checks if at least one of the conditions is true. This is useful when you want to run a block of code if either one of the conditions is met. In this case, you can apply certain logic to both the parent and child processes based on the outcome of the conditions.

Q3: Write a C++ program that uses `fork()` to create a child process. Use an `if-else` statement.

```
Loading...

Welcome to Fedora 33 (riscv64)

[root@localhost ~]# nano forkProgram.cpp
```

```
  GNU nano 5.3                                  forkProgram
#include <iostream>
#include <unistd.h>
int main()
{
int p=fork();
if(p<0){
cout<<z"Fork Failed"<<endl;
return 1;
}
else if(p==0)
{
cout<<"Hello from child process"<<endl;
}
else
{
cout<<"Hello from parent process"<<endl;
}
return 0;
}
```

```
[root@localhost ~]# g++ -o forkProgram forkProgram.cpp
```

Explanation:

In this program, the `fork()` function creates a child process, and the `if-else` statement is used to distinguish between the parent and child processes. If `fork()` returns 0, it is the child process, and if it returns a positive value, it is the parent process. The `if-else` block ensures that both the parent and child processes can perform different tasks or print different messages.