**Lab on Processes:**

**Intro:**

The purpose of this lab is to analyze how an OS initiates and terminates processes using fork(), and how the OS intersperses output from different processes.

**Method:**

Enter the following two files and execute in the lab:

Program1.cpp Program2.cpp

#include <iostream> #include <iostream>

#include <unistd.h> #include <unistd.h>

using namespace std; using namespace std;

int main() int main()

{ {

cout << “Ha! “ << endl; cout << “Ha! “;

fork(); fork();

cout << “Ho! ” << endl; cout << “Ho! “;

fork(); fork();

cout << “Hu!” << endl cout << “Hu!”;

fork(); fork();

cout << “He! “ << endl; cout << “He! “ << endl;

} }

Compile and run as follows:

$ g++ simplefork.cpp –o simplefork

$ ./simplefork

**Results:**

List the output for both programs below.

Program 1 output:

Ha!

Ho!

Ho!

Hu!

Hu!

He!

Hu!

He!

He!

Hu!

He!

He!

He!

He!

He!

Program 2 output:

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

Ha!Ho!Hu!He!

**Analysis:**

Why are we seeing what we are seeing? What is the operating system doing? Hint: remember that each child is an identical copy of the parent at the time of creation and that the endl flushes the I/O buffer (actually prints it).

In program 1, the program has an endl at the end of each line using containing cout. So, only 1 output will be in the output buffer when endl prints it. This is why each word is in its own line. So, when all fork() methods are completed, the sum of all endl executed results in 15 lines of text. In program 2, the program has just one endl at the end of the entire code. So, each time the fork() is called, it almost acts in an exponential growth. The first fork() doubles the amount of lines printed to 2. The second fork() doubles that amount to 4. And the third fork() doubles that amount to 8.

**Conclusion:**

What did you learn from this?

Well, I did learn to run code in the command line. I didn’t know that you could use linux to compile and run c++ code. So that was cool to learn. I also learned to analyze code and really understand how code will process. The fork() method seems simple to understand, and it is. But there are so many ways to use it and so many ways to implement it. This lab helped me really look at code and break down the processes to determine its output.