CPE 435: OPERATING SYSTEMS LABORATORY.

Lab01

Overview of OS model and Programming Tools.

Submitted by: Dan Otieno.

Date of Experiment: 01/17/23.

Report Deadline: 01/20/23.

Demonstration Deadline: 01/20/23.

Introduction

This lab served as an introduction to operating systems processes by testing various samples of demo codes. This also served to refamiliarize with the Linux terminal commands that will be used widely throughout the course. This demo codes are not resubmitted with this report, however, the code for Assignment # 4 is copied below.

Theory

Not required for this lab since it was mostly introductory in nature. However, most of the demo code testing involved visualizing and understanding concepts of forking in operating system structure.

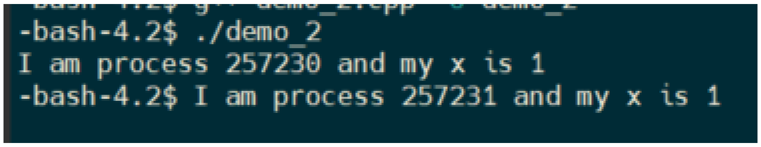
Results & Observation

Assignment 1:

Description:

When running the program, the output showed pid increasing by 1, but each pid had an x value of 1.

Program Output:



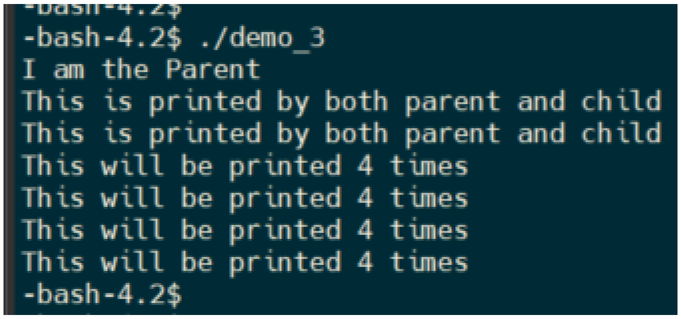
```
-bash-4.2$ ./demo_2
I am process 257230 and my x is 1
-bash-4.2$ I am process 257231 and my x is 1
```

Assignment 2:

Description:

From the output, it looks like there were about 8 processes.

Program Output:



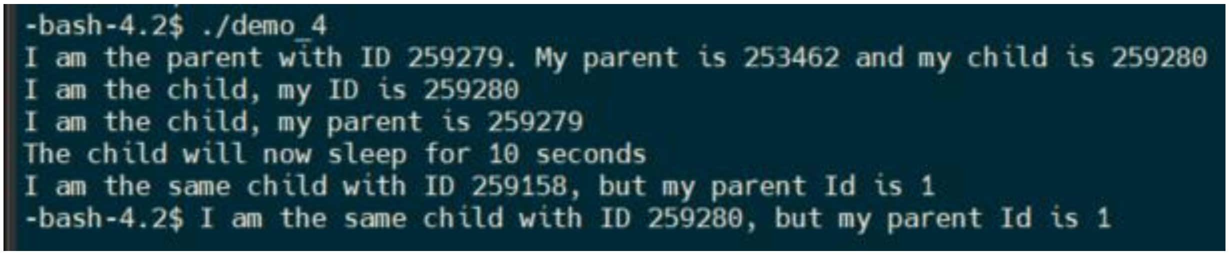
```
-bash-4.2$
-bash-4.2$ ./demo_3
I am the Parent
This is printed by both parent and child
This is printed by both parent and child
This will be printed 4 times
This will be printed 4 times
This will be printed 4 times
This will be printed 4 times
-bash-4.2$
```

Assignment 3:

Description:

An orphan process is one whose parent process terminates before itself i.e. the parent is dead but the child process is still alive. The Pids of the orphan processes in the output below us 259280.

Program Output:



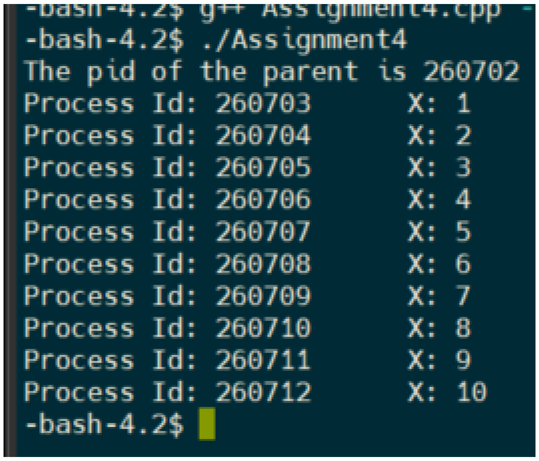
```
-bash-4.2$ ./demo_4
I am the parent with ID 259279. My parent is 253462 and my child is 259280
I am the child, my ID is 259280
I am the child, my parent is 259279
The child will now sleep for 10 seconds
I am the same child with ID 259158, but my parent Id is 1
-bash-4.2$ I am the same child with ID 259280, but my parent Id is 1
```

Assignment 4:

Description:

The program below demonstrates a fork of 10 children, with their own pid and serial number.

Program Output:



```
-bash-4.2$ g++ Assignment4.cpp -
-bash-4.2$ ./Assignment4
The pid of the parent is 260702
Process Id: 260703      X: 1
Process Id: 260704      X: 2
Process Id: 260705      X: 3
Process Id: 260706      X: 4
Process Id: 260707      X: 5
Process Id: 260708      X: 6
Process Id: 260709      X: 7
Process Id: 260710      X: 8
Process Id: 260711      X: 9
Process Id: 260712      X: 10
-bash-4.2$
```

Appendix

Assignment 4 code.

```
using namespace std;
#include <iostream>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>


int main()
{
        pid_t pid, x;

        cout << "The pid of the parent is " << getpid() << endl;

        for(x = 1; x <= 10; x++)
        {
                pid = fork();

                if (pid == 0 ) /* we are the child */
                {
                        cout << "Process Id: " << getpid() << "\tX: " << x <<
endl;
                        exit(0);
                }

        }

        return 0;

}
```