The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-01

Overview of OS model and Programming Tools

Introduction

The goal of this lab is to familiarize you with your work environment for using the operating systems and to get started with linux processes. You will be using the linux machines for assignments in the lab for the entirety of the course. All programs will be demonstrated on these machines.

Part 1: Entering, Compiling and Running a short program in Lab

For editing and preparing programs, the following notes will present an introduction to the *vi* text editor and the GNU C++ compiler. There are several other editors available that you may wish to use instead of *vi*.

Type the following in the terminal:

vi filename.cpp (starts the editor)

i (enter insert mode)

Following is the code that you will insert in your file:

Demo Code

Demo Code 1

```
using namespace std;
#include <iostream>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char * argv[])
{
    pid_t c_pid;
    cout << "The pid of the parent is " << getpid() << endl;
    c_pid = fork();
    if (c_pid == 0 ) /* we are the child */
    {
            cout << "I am the child, my parents pid was " << getppid() << endl;
            exit(0);
    }
    cout << "I am the parent, the child's pid is " << c_pid << endl;

    exit(0);
}
```

Press: [ESC]-colon-wq-[Enter] after you are done typing the code. This will close the *vi* editor and save your work.

In terminal, **g++ filename.cpp -o filename** [Enter]

The above statement tells gnu compiler to compile your program into filename which is an executable

In terminal, **/filename** [Enter] should run the executable

Within *vi*, you can enter command mode at any time (escape-colon). You can move the cursor with either the j-down, l-left, h-right, k-up keys, or sometimes, but not always, with the cursor keys. A good thing to remember is escape-colon-quit-! which gets you out of the editor without saving the file (a good thing when something bad has happened).

To understand any command in linux, you can generally execute the *man* command, for example:
man fork

This should print the man page for fork. If you are not sure of the command you want to use, you might try the command “apropos searchword”, which looks up commands which have searchword in the man page. Alternatively, you might want to buy a book on linux for a few dollars at the bookstore. Old books are generally ok for understanding commands.

Extra Demo Codes

Demo Code 2

The following code forks and prints the value of x from both the processes along with their process id.

```
/* This program illustrates the use of fork */

#include <stdio.h>
#include <iostream>
#include <unistd.h>
using namespace std;

int main()
{
   int x;
   x = 0;
   fork();
   x++;
   // This should be printed twice, once by the parent and once by the child
   cout << "I am process "<< getpid() << " and my x is " << x << endl;
   return 0;
}
```

Assignment 1

Analyze the output for the demo code above in terms of order in which output statements are printed. You may want to run the program a few times. What do you observe?

Demo Code 3

```
/* This program illustrates multiple fork operations */

#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <unistd.h>

using namespace std;

int main()
{
    printf("I am the Parent\n");
    fork();
    printf("This is printed by both parent and child\n");
    fork();
    printf("This will be printed 4 times\n");
    return 0;
}
```

Assignment 2

How many processes are created from the above demo code? Explain your answer.

Demo Code 4

```
/* This program illustrates the death of the parent process before the child is terminated. */
/* The child process is now considered an orphan process since the parent is dead. */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    int pid;
    pid = fork();
    if (pid == 0)
    {
            printf("I am the child, my ID is %d\n", getpid());
            printf("I am the child, my parent is %d\n", getppid());
            printf("The child will now sleep for 10 seconds\n");
            sleep(10);
            printf("I am the same child with ID %d, but my parent Id is %d\n", getpid(), getppid());
    }
    else
    {
            printf("I am the parent with ID %d. My parent is %d and my child is %d\n", getpid(), getppid(),
pid);
            sleep(5);
    }
}
```

Assignment 3

What is an orphan process? Are there any orphan processes in the above code? What is the pid of the orphan process if any are present?

Part II (Expanding the program in Demo 1)

Assignment 4

For homework, write a program that forks off 10 children, each of which will print out its pid and its serial number (1,2,3,4,5...10).

Report Format

- Theory (if asked for in the lab assignment)
- Procedure and Results (what I did −− and what happened when I did it!) Go through the lab step by step and describe what happens as you progress through it. It is a good idea to have another window open where you can record what is going on as it happens.
- Make sure to answer any questions that are asked and perform any procedure that is asked of you. These are often underlined.
- You can capture any output from the program by using the UNIX redirect operator that sends data that normally goes to the screen to a text file. This can be merged into your report.
- Conclusion: Brief statement on what you believe was the major item(s) that you were able to learn and/or demonstrate.
- Source code should be included at the end of the report in separate tables
- The report should be a single PDF file submitted to canvas