CSE 320 Spring 2019 HW #3 Part II

March 25, 2019

Deadline: April 7, 23:59 Stony Brook time (Eastern Time Zone)

1 Introduction

The goal of this homework is to get your hands dirty with more C programming but more importantly, to help you to familiarize yourself with such concepts as creating new processes using `fork()`, using *signals* and how to block them, and how to run programs using `execve()`. This homework consists of three parts.

In the second part, you need to implement a collection of three small programs to get you started and obtain some practice with concepts covered in this homework.

2 Program I

This exercise will help you to get started with signal handlers and `fork()` function.

In the first program, you need to read a number N from the user through the command line arguments and output the last digit on the Nth Fibonacci number. Your application should parse the command line to get the number N and create a child process using `fork()` function. The child process should calculate the last digit of the Nth Fibonacci number and exit with the return value equal to that last digit. The parent process should have a signal handler installed to catch the `SIGCHLD` signal and once it gets the signal to reap the child and print the final output by reading the exit status of the child. The program should finish within one/few second(s) even for big values of N.

You should call the executable "fib" and we should be able to compile your program by issuing the command "make fib". Below you can find an example of a sequence of commands that we will use to run your program:

```
$ make fib
$ ./fib 16
```

And this should print only "7" as it is the last digit of the 16th Fibonacci number.

3 Program II

This exercise will help to get started with `execve()` function.

In the second program, you need to implement a simple shell that prompts a user for the command and tries to execute it unless the command is the "exit" command. If it the "exit" command, then your program should stop its execution. Otherwise, the program should try to execute the provided command using `execve()` function. If it fails then the program should print "Failed to execute CMD" where **CMD** should be substituted with the command that your program failed to execute.

You should call the executable "simple_shell" and we should be able to compile your program by issuing the command "make sshell". Below you can find an example of a sequence of commands that we will use to run your program and an example of the possible execution:

```
$ make sshell
$ ./simple_shell
shell> ls -a
```

And this should have the same effect as typing "ls -a" in your terminal in the same directory.

4 Program III

This exercise will help you to get some experience with `sigprocmask()` function.

In the third part, you need to implement two programs. The first program is a simple shell program called "shell" that needs to run the second program called "child". The shell program needs to ignore the `SIGINT` signal. It should run the "child" program every time the user input "run" command in the shell and then return back to the shell. To exit the program, the user should input the "exit" command.

The second program called "child" should first block `SIGINT` signal and then print numbers one through five inclusive with one-second interval and each number being on a new line. After that, the `SIGINT` signal should be unblocked and then the program should continue with printing numbers six through ten inclusive with one-second interval and each number being on a new line. After finishing printing, the "child" program should exit.

You should call the executables "shell" and "child". We should be able to compile your programs by issuing the command "make sigblock". Below you can find an example of a sequence of commands that we will use to run your program:

```
$ make sigblock
$ ./shell
shell> run
```

5 Note

No zombies. No memory leaks. No crashes.

6 Requirements

You need to follow the naming conventions and the asked way to implement these exercises. If you are asked for some output then you need to print only what was asked and nothing else.

7 Submission

In the third part of the homework, you will be provided with the link to create the GitHub repository and specific instructions on how to submit this part of this homework.