Trent University

COIS3380H

Lab 5

Due: March 24th, 2023

Lab #5 – Signal Processing

In this lab, you will demonstrate your knowledge of signal processing in Unix (*signal* or *sigaction*). You are required to write a C program that contains ***three*** signal handler functions:

Your program should create a counter variable. Counting details in part 1. When the program finally completes, the program should output the number of times the user depressed CTRL-C.

(1) The first is to handle the **SIGINT** (CTRL-C, ^C) signal. This signal handler should increment the counter.

> The signal handler should also contain a MAX value (say 5) that is used. In the case that the maximum is reached:
>
> - a message indicating that the MAX has been exceeded should be output
> - the signal handler should be "deactivated" so that the **next** instance of the ^C signal causes the program to terminate.

(2) The second signal handler should deal with the **SIGQUIT** signal (CTRL-\, ^\). If the user sends this signal, the signal handler should create a new process (*fork*).

> This child process should print a message and then send a SIGUSR1 signal (using *kill()*) to its parent. The child should terminate with no error code. This must be accomplished without using any of the exec() family of calls.
>
> The parent side of the fork() should just exit the signal handler without doing anything.

(3) The final signal handler (for the parent) should catch the **SIGUSR1** signal (sent by the child process in part 2). This handler should print a message that the program is over and then exit. The parent should simply **wait** for the child to terminate, then exit gracefully.

The main program should contain a loop **forever waiting for a signal** (i.e. pause).

Sample output of your program might look something like:

```
jacques@loki:~]$ ./lab5
Wait for another signal ...
^CThis is the 1 time you pressed ctrl-c
Wait for another signal ...
^CThis is the 2 time you pressed ctrl-c
Wait for another signal ...
^CThis is the 3 time you pressed ctrl-c
Wait for another signal ...
^CThis is the 4 time you pressed ctrl-c
Wait for another signal ...
^\I am the child and I am sending a signal
You pressed ^c 4 times.
Child sent a signal so I guess you are bored, have a great day!
```

You are required to hand in **well document** and modular programs plus sample output showing the functionality of your.

REMEMBER: There are a number of ways your program can finish executing. Your LOGS should contain a demonstration of **each**. Zip all of your code, using the proper naming convention and directory paths, testing and sample results into a single submission for Blackboard.