COMP3000 - Exercise 5 (Individual)
Signal - Thread

February 6, 2018

1. Create the program of a process with the following behavior:

   (a) The process forks a child process.

   (b) The parent process sets a handler for the signal `SIGUSR1`. It pauses. When signal `SIGUSR1` occurs, it prints the message `SIGUSR1 was raised!` and terminates.

   (c) The child process gets the parent process ID. Using the system call `kill`, the child process sends the signal `SIGUSR1` to the parent. The child terminates.

   The program must be asynchronous signal safe. Verify for the presence of errors when `fork` and `kill` return. Use `errno` and print a relevant error message, if applicable. While debugging your program, you may have to force the termination of a process. For that purpose, use the Linux command `kill -KILL pid`. Assuming this program is named `evilchild`, the execution should look like this:

```
$ ./evilchild
SIGUSR1 was raised!
```

2. Using the file `primes.c`, build a program that

   (a) generates two random integers i and j (not greater than 5000),

   (b) prints i and j,

   (c) starts one thread that computes the i-th prime number p_i,

   (d) starts one thread that computes the j-th prime number p_j,

   (e) waits until both threads terminate and

   (f) prints p_i, p_j and their product.

   The execution should look like this:

```
$ ./primes
i is 1956, j is 1727
pi is 16963, pj is 14747
their product is 250153361
```

Your solution must not cast a void pointer (a 64-bit value) to an int (a 32-bit value), and vice versa.

Due date: February 18. Submit you work on cuLearn. This exercise must be done in the C programming language under Linux kernel 4.4 (Ubuntu 16.04 has it). Submit a single *tar.gz* file. Submit source code and makefiles. You are responsible for the completeness of your submission. You are responsible for submitting your work on time. Submissions that do not compile are not accepted.