Lab 3 - Description

(Signal Processing in C)

Lab Overview:

For this lab, we will be learning how to process signals in C. A signal is a special integer that the operating system uses to communicate directives to its processes. Consider an example we are familiar with: `SIGSTERM.` This is the signal that is sent to a process to terminate it. You can see for yourself if you start a program then enter CRTL-C. This lab will focus on how to send signals to our child process to accomplish specific tasks.

Core Tasks:

1. Add a process pool to the main (see project 1 part 1 pseudo-code)
2. Create a handler function for the `SIGUSR1` signal in the child.
3. Create a signaler function in the parent.

Task Details:

1. Add a process pool to the main.
    a. See project 2: Part 1.
    b. This process pool should spawn 5 processes.
    c. A file **lab5.c** has been provided (see attached files), use this file.
2. Create a handler for the `SIGUSR1` signal in the child.
    a. This handler will intercept and deal with `SIGUSR1`.
    b. Your handler **must** do the following:
        i. When the signal is given, print the following:
            1. **"Child Process: <pid> - Received signal: <signal>"**
        ii. After the child receives the signal it calls **sigwait()** before the infinite loop**.**
            1. http://man7.org/linux/man-pages/man3/sigwait.3.html
        iii. If it receives the signal again the child will exit.
3. Create a signaler function in the parent.
    a. The signaler function will take the process pool and a signal as parameters.
    b. You signaller **must** then do the following:

i. Send the `SIGUSR1` signal to each child. After this resend the signal to restart the processes.
ii. After this, it will send the `SIGINT` signal to all children to terminate them.

Remarks:

Here are some resources to help you with this lab.

1. To register a signal handler use **sigaction(2)**:
    a. http://man7.org/linux/man-pages/man2/sigaction.2.html
2. To send a signal use **kill(2)**:
    a. http://man7.org/linux/man-pages/man2/kill.2.html

Submission Requirements:

In order to receive any credit for a lab, completion of the labs' core tasks must be demonstrated to the TA's. In order to receive points for this lab the student must do the following:

1. Compile and run your code (we must see it compile and execute successfully).
2. Valgrind output with leak-check and mem-check showing no memory leaks.
3. Submit your file to Canvas.
    a. The submission must take place before the end of the lab. Canvas will lock the submission after your lab time. If you are switching your lab day (perfectly fine), you must tell us before your assigned day.