Lab 5 - 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. Write a main function that spawns 5 processes.
2. Make each process wait before the exec call.
3. Create a signaler function that sends the given signal to every child process.

Task Details:

1. Write a main function that spawns 5 processes.
    a. This process pool should spawn 5 processes and save them in an array.
    b. Each process will run the attached iobound.c file.
2. Each child process will wait until it receives the SIGUSR1 signal.
    a. Use **sigwait(3):** http://man7.org/linux/man-pages/man3/sigwait.3.html
    b. Before waiting the child must print the following:
        i. **Child Process: <pid> - Waiting for SIGUSR1…**
    c. After the child receives the signal print the following:
        i. **Child Process: <pid> - Received signal: SIGUSR1 - Calling exec().**
3. Create a signaler function that is called from the parent process.
    a. The signaler function will take the process pool and a signal as parameters.
    b. Your signaller **must** then loop through each child and do the following:
        i. Sleep for 1 second. See the sleep function mentioned in lab 4.
        ii. Print: **Parent process: <pid> - Sending signal: <signal> to child process: <pid>**

iii. Send the signal to each child. Using kill(2): http://man7.org/linux/man-pages/man2/kill.2.html

4. Send signals to the children.
    a. First, we will send the SIGUSR1 signal to all children.
    b. After this, we will send the `SIGINT` signal to all children to terminate them.

Remarks:

We strongly recommend that you get started on this lab early and that you visit our office hours or post on the discussion board if your have any questions. Of course, if you are finished with part 2 of the project you can add the above-mentioned print statements and show us that as well.

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.
    a. Valgrind output to the console is fine. Fist compile your code with the -g flag then run the following: **valgrind --leak-check=full --tool=memcheck ./a.out**.
3. Submit your lab 5 folder as a tar.gz to Canvas.
    a. The submission must take place before the end of the lab period.