Tutorial 10
Processes and Threads

Learning Objectives

After this tutorial, you will be able to:

- Use fork to create child processes
- Use threads to distribute computationally intensive operations

Tutorial

This week's tutorial will modify the `send` / `handle` program from last week so that it becomes a parent / child process with the parent sending signals to the child.

In order to receive full marks, you must complete everything in **both** of the following tutorial sections.

Tutorial - Forking

1. Download the `T10.tgz` from the tutorial module on Brightspace. Extract and read through the tutorial files.

2. Write a program in `t10-fork.c` which creates a new process in `main()` using the `fork` system call. Using the return value from `fork()`, print out whether the process is the child or the parent.

3. Add signal handlers to *only* the child process so that it prints to `stdout` when it receives `SIGUSR1`, `SIGUSR2`, or `SIGINT`. The child should then wait until receiving `SIGINT` before terminating.

   Hint: The child's code should look very similar to the code for handle *from Tutorial 9.*

4. Add code for (only) the parent process so that it prompts the user to choose which signal to send to the child, then send it. The process should terminate once it has sent a `SIGINT` to the child process.

Tutorial - Threading

5. Speed up `t10-threads` by using multiple threads to determine if the ten numbers are prime. You should not make any changes to the `prime()` function itself.

Saving and Submission

In-Person Attendance: Submission is optional, as you can be checked off in class, but you must still complete the required portion of the tutorial for marks.

Asynchronous: If you are completing the work on your own at home, make sure to follow the submission requirements.

1. Create an archive to submit your files, and include all of the files that you worked with in the tutorial.
    a. They may be submitted as a .tar, .tar.gz, bz2, or .tgz file
    b. Make sure to include all of the code needed to run your program

2. For **full marks** you should have completed all problems in the Tutorial section (and they should be seen upon running the program).

3. For **part-marks** you should have attempted to complete most of the tutorial. Grades for part-marks will be at TA discretion.