COMP 2401 -- Tutorial #9

Sockets and Threads

Learning Objectives

After this tutorial, you will become familiar with:

- **sockets**
- **threads**, and
- **semaphores**

Tutorial Steps

1. Sockets
   The pseudo code on p199 of Chapter 5 lays out the steps for setting up a server using sockets:

```
Open the socket
Bind the socket
Listen on the socket
while (true) {
      Accept a socket request
      while (client has not "hung up" yet) {
            Receive the buffer from the client
            Process the request
            Send a response to the client
      }
      Close client socket
}
Close server socket
```

   Read the corresponding client-server example on pages 200-203 of Chapter 5, and then download it from the code part of Chapter 5 (`server.c` and `client.c`).

   Compile, run and observe the interaction between the server and the client. Try running a few clients in a sequence without stopping the server. Observe that the server process keeps going, waiting for clients to connect even when no clients are running. How do we stop it other than explicitly killing it from the shell?

2. Change `server.c`
   Change `server.c` to track the number of clients it served during its lifetime and print how many it served once it shuts down.

3. Threads
   Threads provide a different mechanism for concurrency than processes (see p209).
   What is the main difference between threads and processes?
   Read the `thread.c` example on p210. Download, compile and run it.

4. `badThread.c` (p214)
   Download, compile and run it. What is wrong?

5. Semaphores to the rescue
   In the `badThread.c` example the variable count is incremented by both threads. The way to make sure that only one thread at a time increments the count is to use a semaphore as in `semaphore.c` (p216). Download and try it. The result now is correct.

6. Speeding up `prime.c`
   Get `prime.c` from the T9 folder. Speed it up by making each prime call a thread and call the new program `primeThreads.c`
   Note the difference in execution time by using the time command from the shell:

```
> gcc –o prime prime.c –lm
> time ./prime
        (Note the execution time)
>gcc –o primeThreads primeThreads.c –lm –lpthread
>time ./primeThread
        (Is there a difference with the time above and if so, why?)
```