CS4023 – Lab Exercise, Week 10

POSIX Threads (pthreads) on Linux

The goal of this exercise is to see how the **`pthread`** library can be used on Linux for creating and synchronizing concurrently running threads.

Step 1. Download **`example4.c`** from the module's website.

Step 2. Open **`example4.c`** (with emacs, for example). This code creates two threads which modify a shared global variable **`counter`**.

Step 3. Execute **`gcc -lpthread example4.cc`** at the command line to build the executable **`a.out`**. Then run the executable, i.e. execute **`./a.out`** at the command line.

Note: You need to refer to the pthread library when you build an executable because it is not considered a standard C library. The Linux implementation of pthreads will ultimately use clone() to create Linux threads. However, it is better in your code to use pthreads instead of clone() directly because of two main reasons: clone() is specific to Linux and makes your code less portable; pthreads are easier to work with because the pthread library provides many tools for threads.

Q1. What do you observe when you run **`a.out`**?

Q2. Use the Linux manual pages and the Web to explain the meaning of the parameters of the function **`pthread_create()`**. Explain also **`pthread_join()`** and its parameters.

Hint: Use the online tutorial
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Step 4. The two threads are using a pthread mutex called **`mutex1`** to synchronize their critical sections. Remove all lines with references to **`mutex1`** from the source code and build the executable again.

Q3. Do you observe any difference in the output generated by the modified source code? If yes, what is the reason?

IMPORTANT: In your own time write a report that describes your answers to Q1-3. This will become part of your end-of-semester project.