CPSC 213: Assignment 9

Due: Sunday, November 28, 2010 at 6pm.

Goal

In this assignment we extend the uthread.c package to include synchronization. The new version of uthread.c includes a complete implementation of Spinlocks and Semaphores and a partial implementation of Monitors and Condition Variables. You will complete the implementation of Monitors this week, and Condition Variables next week. It is highly recommended that you get started on Assignment 10 as soon as you finish Assignment 9.

Notes

The uthreads package runs on Intel x86 machines running Linux, MacOS or Cygwin. You can use the department linux machines by connecting to lulu.ugrad.cs.ubc.ca (or lin01, lin02…).

To compile on Linux or Cygwin it is necessary to explicitly include the pthread library by adding "-lpthread" to the gcc command line.

Requirements

Here are the requirements for this week's assignment.

1. Read and comment the implementations of Spinlocks and Monitors inside the uthread.c file.
2. Compile and run the racedemo.c file to see a demonstration of a classic race condition. Compare the behavior in a single-threaded program, and a multi-threaded program when there is no synchronization and when there is. Look at the source code to see how the synchronization is done. What was the problem in the non-synchronized program?
3. Implement a version of a Monitor, called a "Multi-Reader-Single-Writer" Monitor. This Monitor has three operations: **`enter_for_reading, enter_for_writing, and exit`**. The monitor allows multiple threads to enter the monitor for reading, but only a single thread for writing. You will likely need to make changes to the **`uthread_monitor struct`** to keep track of the readers and writers.
4. Test your new monitor using a program with four reader threads and one writer thread competing to enter a single monitor. The monitor will serve to protect an integer counter in your test program. The writer simply increments the counter. The readers read and print the value of the counter. Each thread loops to enter the monitor, reads or writes to the counter, exits the monitor and then yields with probability 0.5. You can compute the probability using the UNIX **`random()`** operation (see its man page).

Material Provided

The files uthread.h, uthread.c, and racedemo.c are provided in the file code.zip.

What to Hand In

Use the handin program. The assignment directory is **a9**.

1. A version of uthread.c with comments for Spinlocks, Monitors, and with the implementation of the Multiple-Reader, Single-Writer Monitor.
2. Your program that tests Multiple-Reader, Single-Writer Monitors and a description of the results of your tests.
3. Answer to requirement #2 above. What was the problem in the multi-threaded non synchronized code (be detailed about what problems can happen and under what conditions they can happen)? What construct did we use and how was it used to fix the problem?

Reminder: Start assignment 10 early!