# CPSC 213: Assignment 10

Due: Sunday, December 5, 2010 at 6pm.

## Goal

In this assignment we continue extending the uthread.c package to include Condition Variables. The new version of uthread.c includes a complete implementation of Spinlocks, Monitors, and Semaphores. You will implement the Condition Variables functions and write a bounded queue and a test program.

## 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. Implement condition variables. Their state is stored in the **`struct uthread_cv`** and they have four operations **`uthread_cv_create`**, **`uthread_cv_wait`**, **`uthread_cv_notify`**, and **`uthread_cv_notify_all`**.
2. Run some simple tests for your condition variable implementation using a single processor (using uthread_init(1)).
3. Implement a bounded queue of integers using a fixed-sized array. The queue will be shared among multiple threads and so must be synchronized - use Monitors to achieve this. Since this queue has a fixed size, it is possible that an enqueue operation will find the queue full and thus have no room at add the new element; use a Condition Variable to block in this case until there is room for the new element. Similarly, a dequeue operation might find the queue empty; use a Condition Variable to block in this case until there is an element in the queue for dequeue to return. Have a look at Unit 2c slides for a queue implementation.
4. Test your implementation using a single processor and four threads: two "producers" that loop enqueueing integers and two "consumers" that loop dequeueing integers and printing them. Create the threads is such a way that both producer threads and both consumer threads get to run during the test.
5. Test your implementation with 2 and 4 "processors" by changing the argument to **`uthread_init()`**. If you can run this on a real multi-processor (e.g., a dual-core CPU) that is great. But, you can also run the multi-threaded version on a uniprocessor. In this

case, the multiple kernel threads created in **`uthread_init`** will be multiplexed across the single processor by the operating system using its scheduling policy (i.e., preemptive, round-robin) which provides a sufficient emulation of a true multi-processor for testing purposes. Explain the differences you see among the two multi-processor executions and the uni-processor execution from question 5.

6. Read and comment the implementation of Semaphores in uthread.c.

## Material Provided

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

## What to Hand In

Use the handin program. The assignment directory is **a10**.

1. A version of uthread.c with comments for Semaphores and with the implementation of Condition Variables (with comments).
2. Your implementation of the bounded queue from requirement 3, and
3. The associated test program from requirement 4 using the bounded queue.