Concurrent Programming

COMP 409, Fall 2010

Assignment 2

Due date: Monday, November 1, 2010
6pm

Assignment Questions

1. In class you were given an implementation of condition variables in Java. That design provided standard SIGNAL-AND-CONTINUE semantics. Modify the code to implement SIGNAL-AND-URGENT-WAIT semantics. **10**

2. Suppose you want to run the following loop in parallel with 2 threads, one executing the even iterations, and one the odd iterations. There are dependencies between the loop iterations, however, and so you cannot just let the threads run without some kind of synchronization. (Note that $f(i)$ and $g(i)$ do not introduce further dependencies but are expensive, and should not be executed more often than necessary).

   Using only semaphores for synchronization, give psuedo-code including two modified loops (one for each thread) that ensures a correct computation. Try to use as few semaphores as possible. **10**

```
for (int i=0;i<100;i++) {
  a[i] = f(i);
  if (i>1) {
    b[i] = b[i-1]*c[i-2];
  } else {
    b[i] = g(i);
  }
  c[i] = a[i+3] + b[i];
}
```

Programming Questions

You may use Java, or PThreads (C/C++) for the following implementations. In all cases your code must be in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not! All programs should include demonstrative, but not excessive, output. Your programs should not have race conditions and should maximize the ability of threads to execute concurrently.

3. In an $(n+1)$-D universe objects move in n-space and (forward) in time, but may exist for different time periods. If we want to model multiple objects moving around within different time frames, however, there are some consistency concerns: it is important to ensure that an object in a future time does not observe something that did not happen in a past time.

   Two objects can act independently if they are in the same time (ignore space conflicts) or if they are distant enough from each other than they could not possibly influence each other in the amount of time in which they differ. This gives each object a "sphere of influence" as a function of time—an n-D sphere corresponding to all the locations an object could possibly reach within a given amount of future time. In a simple form that is easy to compute—the radius of the sphere is just the maximum speed of the object multiplied by the amount of time.

   We can use these spheres to determine whether two objects may conflict. If A is at time t_A and B at time t_B, and we assume wlog that $t_a < t_B$, then B conflicts with A if sphere(A,$t_B - t_A$) intersects with B. If there is a conflict, the later object (B here) cannot advance in time until there is no longer a conflict.

Simulate this design using monitor-based synchronization, ensuring that threads which cannot move are blocked (waiting) until they can move. As the base environment use a bounded portion of 2D space, shaped as a torus for simplicity (ie a 2D screen, wrapping movement at edge boundaries, both left/right and up/down). Create m threads representing objects at random locations and random times (0–10000ms in the future) and give them random 2D trajectories; they all have the same speed s. Time and position should be advanced (if possible) and compared discretely (in 10ms granularity), but also independently for all objects. Once each object reaches time 20000 it should stop moving and that thread should exit.

Keep a count of the number of actual moves made by each object (avoid introducing a sequential bottleneck) and after all threads have terminated output the average number of moves made by your m objects. Try $m = 10$ and different values of s; how does object speed affect your output? Explain why your code does not deadlock. **20**

What to hand in

For assignment submission you will use `moodle`:

`http://moodle.cs.mcgill.ca/moodle`

If you did not use this for assignment 1 be sure that you login and familiarize yourself with the system before the deadline. Clock accuracy varies, and late assignments will not be accepted without a medical note: you have ample time, **do not wait until the last minute** to do the assignment! Assignments must be submitted on the due date **before 6pm**.

Where possible hand in only **source code** files containing code you write. Do not submit compiled binaries or .class files. For the written answer questions submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files.

Note that for written answers you must show all intermediate work to receive full marks.

This assignment is worth 10% of your final grade. **40**