Spring, 2014 ECEN 5033, CSCI 7000-007

Concurrent Programming

Homework 4

Assigned: 04/08/2014; Due: 04/22/2014, before class.

Homework submissions: by email to the TA (gowtham.ramkumar@colorado.edu).

Problem 1

A savings account object holds a nonnegative balance, and provides `deposit(k)` and `withdraw(k)` methods, where `deposit(k)` adds ***k*** to the balance, and `withdraw(k)` subtracts ***k***, if the balance is at least ***k***, and otherwise blocks until the balance becomes ***k*** or greater. `getbalance()` gives the current balance.

1. Implement this savings account using locks and conditions (use any of the Readers-Writers lock from chapter 8). Test by using the 3 functions.
2. Now suppose there are two kinds of withdrawals: ordinary and preferred. Devise an implementation that ensures that no ordinary withdrawal occurs if there is a preferred withdrawal waiting to occur.
3. Now let us add a `transfer()` method that transfers a sum from one account to another:

```
1    void transfer(int k, Account reserve) {
2        lock.lock();
3        try {
4            reserve.withdraw(k);
5            deposit(k);
6        } finally lock.unlock();
7    }
```

We are given a set of 10 accounts, whose balances are unknown. At 1:00, each of n threads tries to transfer $100 from another account into its own account. At 2:00, a Boss thread deposits $1000 to each account. Is every transfer method called at 1:00 certain to return?

Problem 2

Consider the following conditions: An enqueuer waiting on a full-queue or a dequeuer waiting on an empty queue sleep indefinitely, unless woken up by another thread. A thread must send a signal ONLY when it adds an element to an empty queue or removes an element from a full-queue.

1. Implement a bounded partial queue by using a signaling mechanism (your own scheme) that signals to only one waiting dequeuer or enqueuer, and ensure that the lost-wake-up problem does not happen.

2. Implement the same using a signaling mechanism that signals to all waiting dequeuers.

3. Do a performance comparison using timing analysis. Which works faster?