CSE421 Introduction to Operating System
Fall 1999
Project #2
Introduction to Multithreaded Programming

Bina Ramamurthy

October 4, 1999

1 Objective

To familiarize the students with:

- Thread creation and control (suspension, resumption, exit, join, termination, and synchronization).
- Solving simple problems using multi-threads and compare the performance with various degrees of concurrency.
- Design and implement a new primitive called *barrier* for solving certain kind of synchronization problem among threads and
- Object oriented programming by implementing the *barrier* as a *class*.

2 Problem Statement

This project will also be developed in several small steps to help you understand the concepts better.

1. (20 points) Barrier is synchronization primitive which is often used in the realizing the meeting (and synchronization) of many threads at a certain point before proceeding with the rest of the processing. Solaris thread library does not provide a barrier synchronization primitive. Design and implement a class *barrier_t* for synchronization of many threads at a rendezvous point. Observe that this point is within each thread and may not be in the creator. So it is possible that the threads may want to continue processing after successful completion of this "rendezvous". Implement the barrier primitive using the *pthread_mutex_t* and *pthread_cond_t* combination of the primitives provided by the Posix thread library. Test it with just two threads first meeting among themselves, before joining the creator.

2. Even though barrier has a wide variety of applications in networking, AI and graphics, you will implement a very simple application. Consider a temperature measuring experiment which is carried out it two stages: first ten probes (threads) are sent in to measure the temperature of an item in a process control system. Assume that each probe returns the temperature measurement after a steady state is reached. So each probe may take different times to return. Simulate the time taken by putting to sleep (sleep(n)) each thread for a random time between 1 and 60 units. From this preliminary set, the largest two and the smallest two probes are eliminated (cancelled) and the experiment is repeated with the six probes. The readings from the second stage of the experiment are averaged to get the accurate measurement of the temperature.

   (a) (15 points) Implement the above without **Barrier primitive** that you designed but using *pthread_joins*.

   (b) (15 points) Implement it using the Barrier primitive that you wrote.

(c) (50 points) Consider a peoplemover system such as the one you may find in a Disney world or a tram system such as the one in Toronto downtown. The people-mover system can be modeled as a grid of paths with movers (we will call them trains) shuttling people on them. Here is a model of one:

i. Assume that there are two paths in the EastWest(EW) direction and three paths in the North-South(NS) direction.
ii. The EW path is 15 miles long and NS is 25 miles long. The trains move at 60 miles per hour.
iii. Typically one of the tarins in the EW is used for trasporting people from East terminal to West and the second path for the other direction.
iv. On the NS direction two trains are used for moving people to South and one larger (twice) capacity to move people to North.
v. Initially assume that all move at the same speed.
vi. All trains except the larger capacity double decker that is going from North to South, can carry maximum of 100 passengers.
vii. Passengers arrive at random intervals, when a train is at least half full the train can start. If the passengers arrive when there is train they wait is a queue.
viii. Passengers arrive a group of 4 to 10.
ix. The EW paths cross the NS paths. You have to make sure no trains collide at the intersections. In other words, the intersections are to guarded for mutual exclusion.

Simulate this problem using posix multi-thread library. Each train can be implemented using a thread. You may need other supporting thread for "passenger generation".
Expected output :

i. A trace of the events taking place for each train.
ii. Periodic report on number of passengers moved.
iii. A summary at the end of sufficient number of hours representative of the operation of the system.

3 Due Date

The due date : 10/28/99 before midnight.