CSE421 Introduction to Operating System
Spring 2008
Project #2
Multi-Threaded(MT) Programming: Mutual Exclusion and Synchronization

Bina Ramamurthy

February 25, 2008

1 Objective

To familiarize the students with:

- Thread creation and control (exit, join, termination, and synchronization).
- Designing and implementing a multi-threaded application.

2 Problem Statement

1. (25 points) **Sleeping Barber problem** Implement a solution to the sleeping barber problem. Implement the solution using synchronization primitives (pthread_mutex_t for mutual exclusion) provided by the POSIX threads. Your implementation will consist of a control program that (i) initializes the synchronization variables and (ii) creates and terminates the threads for the customer and the barber. Print appropriate message to indicate the events happening. (It is possible messages may be appearing out of order, so prefix the messages with some order related numeral.) Let the control program simulate all possible conditions: Chairs (n) free, chairs full, barber sleeping, customer leaving, customer waiting. Update the basic single barber implementation to multiple barber implementation. Also simulate the single barber version by placing signal from customer to barber ("CtoB") out of the critical region, after the mutex. Observe and note what happens.

2. (30 points) **Multi-processor Synchronization** Larry, Moe, and Curly are planting seeds. Larry digs the holes. Moe then places a seed in each hole. Curly then fills the hole up. There are several synchronization constraints:

   (a) Moe cannot plant a seed unless at least one empty hole exists, but Moe does not care how far Larry gets ahead of Moe.

   (b) Curly cannot fill a hole unless at least one hole exists in which Moe has planted a seed, but the hole has not yet been filled. Curly does not care how far Moe gets ahead of Curly.

   (c) Curly does care that Larry does not get more than MAX holes ahead of Curly. Thus, if there are MAX unfilled holes, Larry has to wait.

   (d) There is only one shovel with which both Larry and Curly need to dig and fill the holes, respectively.

   Design, implement and test a solution for this IPC problem, which represent Larry, Curly, and Moe. Use semaphores as the synchronization mechanism.

3. (35 points) **Thread Scheduling** Write a scheduler for user level threads that works on round robin policy. A controller (factory) creates as many threads as required, it employs a timer that interrupts the scheduler at predetermined intervals to switch the currenlty running thread to a thread in front of a wainting queue. Parameters such as time interval and execution time for the scheduled thread have to be passed into a controller that coordinates the scheduler and a timer that maintains the intervals. You can use alarms and signals to control the threads.

3 Material to be Submitted

1. Submit the **source code** for the programs. Use meaningful names for the file so that the contents of the file is obvious from the name. You may zip all the source files into a single file. Also provide a Pr2README file that explains the contents of the zip file. Pr2README file should have an observation section for each of the four problem. Use tables whereever suitable (10 points for documentation).
2. Use internal documentation to explain your design.
3. Test runs: It is very important that you show that your program works for all possible inputs. Submit **a single script** that shows for each program the working for correct input as well as graceful exit on error input.
4. Submit your makefile.

4 Due date

3/29 submit on-line before mid-night.