CS4328 – Operating Systems
Program Assignment #1 (Total Points: 150)

Due 1 second before class begins on Dec. 2, 2013

Overview

The purpose of this project is to practice process synchronization. There are two independent parts of this project. Please read instructions carefully.

You must submit your project in one zip file (named as firstname_lastname_proj2) to the Project Turn In Site (https://hwupload.cs.txstate.edu/), which should include all your source code for part1 and part2. The submission system will automatically record your submission time, which will be used to decide whether or not you miss the deadline. There will be a penalty of 20% per day (including weekend and holiday) for late submission. The late submission after two days (including weekend and holiday) will not be accepted. Therefore, DO NOT DELAY. Start writing the program from Day 1. If you wait until the night before the due date, you will have a miserable night and it is less likely you could complete the project.

It is your responsibility to ensure that your code can compile and run. You will lose 50% of your total points if your code has compiler or runtime errors (e.g. segmentation fault).

You are encouraged to discuss with other students but you have to **DO YOUR OWN WORK**. You may get ***ZERO*** for this project or ***FAIL*** the class if you copy code from others or allow others to copy your code.

Part 1: The Producer-Consumer Problem with Bounded Buffer (75 points).

In class, we discussed a semaphore-based solution to the producer-consumer problem using a bounded buffer. In this project, you will write the **Pthread** code to solve this problem using three semaphores: empty, full and mutex.

You can assume the BUFFER_SIZE is 10 and you can use a circular queue to simulate the BUFFER. The buffer will be manipulated with two functions, insert_item() and remove_item(), which are called by the producer and consumer threads, respectively. The insert_item() function will insert a random generated integer (between 0 and 1000) to the buffer and the remove_item() function will remove one integer from the buffer. Busy waiting will apply if a producer or a consumer thread cannot enter its critical section.

The main() function will initialize the buffer and create the separate producer and consumer threads. Once it has created the producer and consumer threads, the main() function will sleep for a period of time and, upon awakening, will terminate the whole program. The main() function will be passed three parameters via the command line:

1. How long to sleep before terminating the program (in second).
2. The number of producer threads
3. The number of consumer threads

Hint: Refer to the Demo_Semaphore code for semaphore related operations.

Part 2: The Sleeping Teaching Assistant Problem (75 points)

In this project, you will use Pthreads, mutex locks, semaphores and conditional variables (optional) to solve the sleeping teaching assistant problem.

Assume that your instructor hires a teaching assistant (TA) who helps students with their programming assignments during regular office hours. The TA's office is rather small and has room for only one desk with a chair and computer. There are three chairs in the hallway outside the office where students can sit and wait if the TA is currently helping another student. When there are no students who need help during office hours, the TA sits at the desk and takes a nap. If a student arrives during office hours and finds the TA sleeping, the student must awaken the TA to ask for help. If a student arrives and finds the TA currently helping another student, the student sits on one of the chairs in the hallway and waits. If no chairs are available, the student will come back at a later time.

Your main() program should create n student threads and one TA thread, where n is passed in from the command line. Each student thread will alternate between programming for a period of time and seeking help from the TA. If the TA is available, they will obtain help. Otherwise, they will either sit in a chair in the hallway or, if no chairs are available, will resume programming and will seek help at a later time. If a student arrives and notices that the TA is sleeping, the student must wake up the TA. When the TA finishes helping a student, the TA must check to see if there are students waiting for help in the hall way. If so, the TA must help each of these students in turn. If no students are present, the TA may return to napping.

The main() function will sleep for a period of time and, upon awakening, will terminate the whole program. The main() function will be passed three parameters via the command line:

1. How long to sleep before terminating the whole program (in second).
2. The number of student threads

Perhaps the best option for simulating students programming – as well as the TA providing help to a student – is to have the appropriate threads sleep for a period of time (e.g. 1ms). In addition, please make sure to print meaningful messages at the right stage such as:

The TA is helping student *id.*

Student *id* goes back to program.

Student *id* waits for help in a chair.

The TA is sleeping.

The TA is waked up.

Hint: Refer to the Demo_Conditional_Var code if you decide to use conditional variables.