Project 4: Office Hours **CS 300, Hong, 2024 Spring**

Description

A teacher sets up office hours to answer questions relating to course materials and assignments. The teacher meets one student at a time. There are n chairs outside the office where students can sit and wait if the teacher is currently helping another student. The teacher helps students in a first-come-first-serve way. The interactions between students and the teachers are possibly in four cases: (1) When there are no students who need help during office hours, the teacher goes back to own work. (2) If a student arrives during office hours and finds the teacher on own work, the student raises the teacher attention (which the teacher will respond) to ask for help. (3) If a student arrives and finds the teacher currently helping another student, the student sits on one of the chairs and waits. (4) If no chairs are available, the student will go study and come back at a later time.

You will use threads to simulate the interactions between students and the teacher. The synchronization primitives such as mutex locks, semaphores, or conditional variables in pthread library in C can be used to implement a solution that coordinates the activities of the teacher and the students. A FIFO queue can be used for the seats taken by students.

For this project, you will implement two versions of the solution, one uses only locks and semaphores, and the other uses only locks and condition variables. In both versions, no other synchronization primitives are allowed.

Some Programing Details

The project will use two types of threads, the Students and the Teacher. Each student will run as a separate thread. The total numbers of students should be input from command line. The teacher will run as a separate thread. A student thread alternates between studying for a period of time and seeking help from the teacher. If the teacher is available, one student will obtain help. Otherwise, they will either sit in a chair or, if no chairs are available, will resume study and will seek help at a later time. The number of chairs should be input from command line. If a student arrives and notices that the teacher is on his own work, the student calls the teacher. When the teacher finishes helping a student, the teacher checks to see if there are students on chairs waiting for help. If so, the teacher continues helping each of these students in turn. If no students are present (all chairs are empty), the teacher returns to his own work.

The code will read four integers from command line. They are: the number of students, the number of chairs, and the left and right of the interval for the random time period, which will be used in mytime (int left, int right). If input is wrong, prompt should be given for try again with correct format.

A Short Report

After finishing your code, answer the following three questions based on your code. The answers should be more than 150 words and less than 300 words. (1) will your solution run into deadlock, explain why or why not. (2) Is your solution fair? (or say, will your solution lead to starvation of a student), explain why or why not. (3) Comparing the two implementations that used either semaphores or conditional variable, which one is your favor, explain why or why not.

Suggestions and Additional Requirements:

(1) To simulate student studying for a while in student threads, and the time teacher providing help to a student in the teacher thread, a sleep function for a random time period should be used. A customized function called int mytime (int left, int right) is provided for this purpose. mytime (int left, int right) produces a random number within a given interval between left and right. It calls rand() for a random number. To have different random number sequences for each execution, one should invoke srand(*z*) to set a proper seed *z* in the main program before any invocation of rand().

(2) Each student thread terminates after getting help **twice** from the teacher. In main program, don't forget join student threads. After all student threads terminate, the main program cancels the teacher's thread by calling pthread_cancel() and then entire program terminates. Sample code for starting multiple threads is given in BB, file name: PC-inputs-main.c. It is modified from a code example from OSTEP for the Producer and Consumer Problem. In the code, example of thread sleeping is also provided, e.g., thread sleeps for a random time after being created. mytime() is used to get the random time. You can build your project from this sample code.

(2.1) The example uses an array for one type of thread, also a loop and sleep after each thread is created. Make sure that when using the random sleep duration, set a short sleep time.

(2.2) You need to call the specific function mytime() (see files mytime.c, mytime.h) to obtain a random time to be the input to the sleep function. The makefile is provided.
(2.3) A student thread sleeps when arrival finding no chairs are available; the teacher thread sleeps when finding no students are there.
(2.3) Feel free to use more functions wherever you see fit. The printouts should be used following the same format given in (3).

(3) In the code, print out necessary info about the activities during the execution of the code.
(3.1) Before sleep, print out "Student (or Teacher) <ThreadID> to sleep X sec;".
(3.2) After wake up from sleep, print out "Student (or Teacher) Id <ThreadID> wake up;".
(3.3) For either the student or teacher thread, before calling mutex locks, or wait on semaphores, or conditional variables, printout "Student (or Teacher) <ThreadID> will call mutex_lock / sem_wait / conditional variables cond_wait <synch variable name> ".
(3.4) For either the student or teacher thread, after calling to unlock mutex locks, or post on semaphores, or conditional variables, printout "Student (or Teacher) <ThreadID> call mutex_unlock / sem_post / conditional variables cond_signal <synch variable name> ".

Compiling and Submission Requirements

1) Name your code as your P4-sem-*YourFirstLastName*.c, P4-cv-*YourFirstLastName*.c, respectively, including comment lines at the beginning to show your full name and the project info.
2) Name your report as *P4-YourFirstLastName.pdf* (.docx is fine too). Include your name and course # in the report.
3) Include lines of #include <pthread.h>, #include <semaphore.h> and #include "mytime.h". If you are compiling multiple files, feel free to compile using provided makefile (as sample). Otherwise, you will use -c to indicate compiling only.
4) Compiling in the makefile uses -Wall -std=c99, which will also be used in grading. Grading uses the cs-intro.ua.edu server. You should try to use the same compiling commands at the cs-intro server to compile your code. Errors leads to points drop.
5) At the start of the code, use comments to write your name, class and the project name. Also add other information about the project.
6) Submit only the source code you write. Other source code will be supplied by TA at grading time.
7) Submit to Blackboard "Proj 4" before deadline. Submit two separate pieces of source code for the two implementations (do not zip), and the report.
8) 10 points are allocated to the submission and compiling.
9) 10 points are allocated to the report.

Delay policy:

Follow the course policy. Each one-day delay will drop 20% points. You should start your projects early to avoid potential last minute issues that cause late turn-in.

Grading will check:

(1) Submission and Compilation; (2) Correctness of the executions based on the clearness of the outputs from your code; (3) Code inspection; (4) Correctness and clarity of your report.