Lab Assignment 3: Synchronization and Mutual Exclusion

Notes

You need to work on this project individually.
This project must be implemented in C.

Objectives

- To get familiar with concurrent programming.
- To better understand handling races, synchronization, mutex, and condition variables.
- Learn about debugging concurrent programs.

Overview

CalTrain (the train running in California, USA) has decided to improve its efficiency by automating not just its trains but also its passengers. From now on, passengers will be robots. Each robot and each train is controlled by a thread. You have been hired to write synchronization functions that will guarantee orderly loading of trains. You must define a structure `struct station`, plus several functions described below.
When a train arrives in the station and has opened its doors, it invokes the function

`station_load_train(struct station * station, int count)`
where count indicates how many seats are available on the train. The function must not return until the train is satisfactorily loaded (all passengers are in their seats, and either the train is full or all waiting passengers have boarded). Note, that the train must leave the station promptly if no passengers are waiting at the station or it has no available free seats.

When a passenger robot arrives in a station, it first invokes the function

`station_wait_for_train(struct station * station)` This function must not return until a train is in the station (i.e., a call to station_load_train is in progress) and there are enough free seats on the train for this passenger to sit down. Once this function returns, the passenger robot will move the passenger on board the train and into a seat (you do not need to worry about how this mechanism works). Once the passenger is seated, it will call the function

`station_on_board(struct station * station)`
to let the train know that it's on board.

Requirements

- Write the declaration for `struct station` in the file `caltrain.h`.
- Write the three required functions (described above) and the function `station_init`, which will be invoked to initialize the station object when CalTrain boots in the file `caltrain.c`.
- You must write your solution in C using Pthreads and its Mutex and condition variables. Note, you should not use semaphores or other synchronization primitives.
- You may not use more than a single lock in each struct station.
- You may assume that there is never more than one train in the station at once, and that all trains (and all passengers) are going to the same destination (i.e. any passenger can board any train).
- Your code must allow multiple passengers to board simultaneously.
- Your code must not result in busy-waiting.

Notes

- This assignment is based on this assignment given at Stanford University: `http://web.stanford.edu/~ouster/cgi-bin/cs140-winter13/problemSet0.php`.
- Download `caltrain.h` and `caltrain.c`, and complete the missing code.
- Download and use `caltrain-runner.c` to test your code.
- You can use the included `Makefile` to compile your code. You will need to type the command `make`.

Deliverables

- Complete source code in **C**, commented thoroughly and clearly.
- A report that describes the following: (1) how your code is organized, and (2) the various mutex and condition variables that you used.
- All deliverables are to be put in one directory named lab3_XX, where XX is your ID and then zipped.
- You need to send your code to cs333f17@gmail.com on October 28 before 8 AM. The subject line should be: "Lab Assignment 3 - SID:xx"
- The assignment will also be discussed in the lab.