EECS 111

System Software

Spring 2019

Project 2: Thread & Synchronization

Due Date (May 12th, 2019 11:55 PM)

Instructions

In this project, you are going to understand thread synchronization and mutex lock and write a code to solve some problem. The first step is to read and understand this project description. Then, you can start coding.

By unzipping the project file you will see the following files in the ***p2_student*** directory:

1. `main.cpp`: This is a top file for this project
2. `p2_threads.h`: This is a header file to handle threads
3. `p2_threads.cpp`: This is a file to handle threads
4. `utils.h`: This is a header file for utility functions
5. `utils.cpp`: This is a file header file for utility functions
6. `types_p2.h`: This is a header file for `Person class`
7. `types_p2.cpp`: This is a source code file for `Person class`
8. `Makefile`: This is a compilation script

You need to complete the following files: `main.cpp`, `p2_threads.cpp`, and `Makefile`. The main objective of this project is to create a program that solves the problem below.

After you complete the `Makefile` file, You should be able to build the project code by using following command: `make`. After build, **you must have `p2_exec` file as an executable.** Also, it is suggested to remove all the compiled object files and executable file with the following command: `make clean`. You can change the source code in the given files. In addition, you can add any new header and source code files for this assignment. But, please make sure that your program can be compiled with '`make`' command and satisfies the descriptions below.

In this project, you have to get 2 arguments. You have to print error message and usage example if you didn't provide the arguments. This is an example for the error message.

```
[ERROR] Expecting 1 argument, but got (X).
[USAGE] p2_exec <number>
```

Here is the description for the problem:

A new boutique opened in Irvine that sales both male and female clothing. The shop has three fitting rooms available for both customers in an aisle. However, customers indicated that they prefer to have stalls specified for their gender. In order to solve this issue, the manager came up with the following plan:

- There is a sign, in front of the aisle, which indicates if the room is `Empty, WomanPresent, ManPresent, or Full`.
- Based on this sign, if there is a woman in any one of the stalls, only women are allowed to enter the aisle. The same would occur for men.
- If a person of the opposite gender wants to enter the fitting room, or if all of the stalls are being used, he/she will have to wait in a line.

In order to test this plan, the manager has asked UC Irvine EECS111 students to create a simulator for her. Interestingly, to test your skills, she has requested to use multithreading in the simulator! To create the simulator, you need to follow some rules:

- You should complete a working program that addresses the above problem and that must compile and run.
- Your program will take two arguments in the following order:
  <each_gender_cnt> <number_of_stalls>

- During run-time, you need to generate a sequence of people based on the input argument. If 10 is the input argument, then your program must generate a sequence of total of 20 people that includes 10 men and 10 women.

- The program must execute until all the people finish using the fitting room.

- You must randomly assign the gender to a person and send that person to the fitting room. After that person has been sent to the fitting room, you must wait for a random time interval (between 1 milliseconds – 5 milliseconds) until you can send the next person to the fitting room. Your program must display the following output:

  - `[01 ms][Input] A person (Man) goes into the queue`
  - `[03 ms][Input] A person (Woman) goes into the queue`

- When a person goes into the fitting room, you must randomly assign the time to stay in the fitting room (3 milliseconds – 10 milliseconds). Your program must display the following output:

  - `[56 ms][Queue] Send (Man) into the fitting room (Stay 4 ms), Status: Total: x (Men: x, Women: x)`
  - `[69 ms][Queue] Send (Woman) into the fitting room (Stay 7 ms), Status: Total: x (Men: x, Women: x)`

- The program must display the following during its execution: the time stamp, the state of the fitting room (empty, occupied by women and if so how many, occupied by men and if so how many), the status of the queue including whether it is empty or not and, if not empty what genders are in queue. The example command line output for each case is as follows:

  - `Format: [Time (ms)][Thread name] Behavior, status of the thread`

- `[12 ms][Queue] Send (Man) into the fitting room (Stay 4 ms), Status: Total: x (Men: x, Women: x)`
- `[12 ms][fitting room] (Man) goes into the fitting room, State is (MenPresent): Total: x (Men: x, Women: x)`
- `[16 ms][fitting room] (Man) left the fitting room. Status is changed, Status is (empty): Total: x (Men: x, Women: x)`

- The above example output is only for the one possible scenario. There may be many additional command line outputs.
- Your program must generate different sequence. In other words, your program must show different behaviors for each run of the program.
- The program must contain the following functions: `woman_wants_to_enter`, `man_wants_to_enter`, `woman_leaves`, and `man_leaves`.
- Your code MUST SHOW the parallel behavior.
- TIP: You must have at least two threads. Except main process.
- Your program should not have a deadlock.

Submission

A drop box folder is provided in EEE website. You need to compress all the files in folder into a single archive .**zip file** (no other format will be accepted) and upload it. The deadline for uploading the files is the project deadline. Since your submissions will be processed by a program, there are some very important things you must do, as well as things you must not do:

1. It is imperative to keep the output format same as what you are asked for.
2. You MUST USE only C++98 standard. If you use any new features like C++11 or C++14, your code will not be graded.
3. Make sure your code can be compiled and run. If we cannot compile your code, then your code will not be tested and graded.
4. Your executable filename must be **`p2_exec`**.
5. Your code **MUST SHOW** a parallel behavior. Otherwise, your code will not be graded properly.
6. Your submissions must contain all your files and directories in **p2_<your student id>** directory (i.e. p2_84733922).
7. You must not change the directory structure. In other words, don't change any file or folder name except the top directory. The top directory name (student id) will be used to track your submission status.
8. Since the input file sizes are large, you MUST NOT include input files in your submission.
9. The codes that the students submit will be compared with each other using a program. If they are similar enough, the program will give us a warning about it. So, you can talk to each other about the project, and visit online resources, but you must write your own code.

Grading

1. (5%) Following the submission format
2. (10%) Compile your code with your Makefile without any problem.
3. (20%) Command line output matches with the description for any type of input.

4. (15%) Randomly generate the input sequence and randomly assign the time to stay in the fitting room
5. (50%) Your program work properly and does not have deadlock.

Note

Even though you can generate correct output, that does not mean that your code consider some extreme cases. You should verify your code with some corner cases as well.

If you have any question regarding the project, please ask it in the discussion session.