EECS 111

System Software

Spring 2017

Project 3: Thread Scheduling

Due Date (Jun 8th, 2018 11:55 PM)

Instructions

In this project, your job is to understand thread conditional variable and write a code to solve thread scheduling problem. By unzipping the project file you will see the following files in the ***p3_student*** directory:

1. `main.cpp`: This is a top file for this project
2. `p3_threads.h`: This is a header file to handle threads
3. `p3_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_p3.h`: This is a header file for `Person class`
7. `types_p3.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` and `Makefile`. The main object of this project is to create a program that solve 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 `p3_exec` file as an executable.** Also, it is suggested to remove all the compiled object files and executable file with the following command before submitting your code: `make clean`.

In this project, you have to use 1 argument. You have to print error message and usage example if you didn't provide the argument. This is an example for the error message.

```
[ERROR] Expecting 1 argument, but got (X).
[USAGE] p3_exec <number>
```

Here is the description for the problem:

The assignment code will create 4 worker threads. Each thread has its own task execution time, period (deadline) and so on. (These part are already implemented). You need to give 1 argument. Based on the argument value, your program uses a different scheduling function (0: FIFO scheduling, 1: Earliest deadline first (EDF) scheduling, and 2: Rate-Monotonic (RM) scheduling). For example, if 0 is input argument, then your program should use FIFO scheduling.

- The program must contain the following functions: `fifo_schedule` and `edf_schedule`.
- You need to understand *pthread conditional variable* to complete this assignment.
- **You can ONLY change main.cpp and Makefile file**. You MUST NOT change the other files.
- Your job is to implement two thread scheduling functions for FIFO and EDF scheduling. RM scheduling is NOT mandatory. You need to read the code and understand its behavior to write your scheduler code in main.cpp.
  - If you complete a RM scheduling function, you will get extra points.
- In your code, except main process (the scheduler), **only one thread can be active**. You can wake up any thread when there are no active threads.
- You need to implement a FIFO scheduling function first. Then check whether the FIFO scheduling satisfies all the deadlines or not.
- After that, you need to implement your own scheduling function based on EDF scheduling. Your EDF-based scheduling function MUST satisfy all the deadlines.

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 **`p3_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 **p3_<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. (5%) Command line output matches with the description for any type of input.
4. (40%) Complete FIFO scheduling function.
5. (40%) Complete modified EDF scheduling function

6. (10%) Complete RM scheduling function.

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.