EECS 111

System Software

Spring 2017

Project 1: Fork & Pthread

Due Date (April 22nd, 2017 11:59 PM)

Instructions

In this project, your job is to understand how to create processes and threads and use multiple processes and threads to process multiple data in parallel.

The first step is read and understand this project description. There are several requirements that you must satisfy to get score. Then, you can start writing your code for this project.

You can unzip the project file by using this command: `tar -xvf ./eecs111-proj1.tar.gz`. Then you will see the following files in the ***p1_student*** directory:

1. `./data/`: This directory includes csv files for input data
2. `./output/`: This directory is for output data
3. `main.cpp`: This is a top file for this project
4. `p1_process.h`: This is a header file to handle processes
5. `p1_process.cpp`: This is a file to handle processes
6. `p1_threads.h`: This is a header file to handle threads
7. `p1_threads.cpp`: This is a file to handle threads
8. `Makefile`: This is a compilation script

You need to work on the following files: `main.cpp`, `p1_process.cpp`, `p1_threads.cpp`, and `Makefile`. The main objective of this project is to generate some statistical results from multiple files in parallel. You need to calculate average, median, and standard deviation for each class and get the sample grade for top, middle and lowest score. Then, the results need to be stored into the output directory.

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 `p1_exec` file as an executable.** Also, it is suggested to remove all the compiled object files and the executable file with the following command: `make clean`.

In this project, you have to use 2 input arguments: the number of processes and the number of threads. You have to print error message and usage example if you didn't provide those arguments. This is an example for the error message.

```
[ERROR] Expecting 2 argument, but got (X).
[USAGE] a1_exec <number of processes> <number of threads>
```

Besides, 2 arguments for the executable, some additional descriptions for your program are as follows:

- For each input file, you need to generate: Average, Median, Standard Deviation, Top 3 students, Middle 3 students, and Lowest 3 students. Calculate the result rounded up to the three digits after the decimal point.
- Each input file has the following data: 1 line for class code (6 digits) and the rest of the lines for student id (10 digits) and grade.
- Your program needs to create multiple processes to handle multiple files in parallel. Each child process handles at least one file. In other words, one child process may handle multiple files.
- Only child process should create multiple threads to perform the calculation. So, process hierarchy will be the following: Main process -> Child processes -> Threads
- You must not waste any resource, thus you should not create any non-working process or thread.
- The result needs to be saved to output directory with the following filename: `<input filename>-stats.csv` (i.e. os-stats.csv).
- When a process is created, and terminated, you have to print the message with process id to indicate the process is created/terminated (you don't need to do this for threads).
- Main process must complete its behavior after its child processes.

Example command line output is as follows:

```
./p1_exec 2 4
Main process is created. (pid: 8456)
Child process has been created. (pid: 3992)
Child process is created. (pid: 4692)
...
Child process is terminated. (pid: 3992)
Child process is terminated. (pid: 4692)
Main process is created. (pid: 8456)
```

Example output file should look like as given below:

```
Class code: 199405

Average: XX.XXX
Median: XX.XXX
Std. Dev: XX.XXX

Top 3 students:
Rank  Student id    Grade
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX

Middle 3 students:
Rank  Student id    Grade
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX

Lowest 3 students:
Rank  Student id    Grade
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX
 XX   XXXXXXXXXX   XX.XXX
```

Grading

Your project code will be automatically graded. There are two reasons for this:

1. A grader program can test your code a lot more thoroughly than a TA, yielding more fair results.
2. An autograder can test your code a lot faster than a TA can.

Of course, there is a downside. Everything that will be tested needs to have a standard interface that the grader can use, leaving slightly less room for you to be creative. Your code must strictly follow these interfaces.
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. Your executable filename must be **`p1_exec`**.
2. You must store all your files and directories in **p1_<your student id>** directory (i.e. p1_84733922).
3. You must not change the directory structure. In other words, please don't change the filename except the top directory. The top directory name (student id) will be used to track your submission status.

Submission

A drop box folder is provided in EEE website. You need to compress all the files of your folder into a single archive file (zip, rar, etc.) and upload it. The deadline for uploading the files is the project deadline.

Tasks

1) (5%) Following the submission format
2) (20%) Compile your code with your Makefile without any problem.
3) (20%) Command line output matches with the description for any type of input.
4) (55%) Output files match with the description for any type of input.

Note

Even though you can generate correct output, that does not mean that your code considers some extreme cases. Please verify your code with some corner cases.

If you have any question regarding the project, please send an e-mail to following address: haeseunl@uci.edu. You MUST cc the professor and TA when you send an e-mail.