EECS 111

System Software

Spring 2019

Project 1: Fork & Pthread

Due Date (April 21nd, 2019 11:55 PM)

In this project, you are going to understand how to create and use multiple processes and threads to process input data in parallel. The first step is to read and understand this project description. There are several requirements that you must satisfy to get score. Then, you can start coding project.

By unzipping the project file you will see the following files in the ***p1_student*** directory:

1. `./input/`: 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 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, you should apply changes as it is required to the `Makefile` and make sure that your program can be compiled with '`make`' command. The result of compiling your codes using '`make`' command should be **`p1_exec` file as an executable.** It is suggested to remove all the compiled object files and the executable file before submitting with the following command: `make clean`.

Here are some descriptions of how your program (p1_exec) should behave:

- `p1_exec` gets 2 arguments: maximum number of child processes and maximum number of threads per each child process.
- 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.
- each input file, you need to first sort the input file with a multithread sorting algorithm (Merge sort is suggested), then calculate the followings statistics: Average, Median, Standard Deviation.
- Only child process should create multiple threads to perform the sorting. 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 results need to be saved to output directory with the following filename: `<input filename>_stats.csv` and `<input filename>_sorted.csv` (i.e. os_stats.csv and os_sorted.csv).
- When a process is created and terminated, you should 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.

Complete source will provide the following output.

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

Complete source will provide the following output files:

os-stats.csv content:

```
Average,Median,Std. Dev
XX.XXX,XX.XXX,XX.XXX
```

os-sorted.csv content: (Here, number of float digits should be same as input file)

```
Rank,Student ID,Grade
 X,XXXXXXXXXX,XX.XXX
 X,XXXXXXXXXX,XX.XXX
 X,XXXXXXXXXX,XX.XXX
 …
```

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 **`p1_exec`**.
5. Your submissions must contain all your files and directories in **p1_<your student id>** directory (i.e. p1_84733922).
6. 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.
7. Since the input file sizes are large, you MUST NOT include input files in your submission.
8. A sample auto-grader script which is written in python is provided for you. You can use it to double check your output format. However, the actual auto-grader script is different. It will use other test benches, will run in a sandbox, and will have a software similarity detection feature. Similarity detection feature will compare your codes together and with the other codes available online. If they are similar enough, it 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. (15%) Compile your code with your Makefile without any problem.
3. (20%) Command line output matches with the description for any type of input.
4. (60%) Output files match with the description for any input.
5. (-25%) If there is no multithreading
6. (-25%) If there is no multiprocessing

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.
- There will be always exactly five inputs with the names currently given to you in the assignment 1 package.
- If two students have the same grade, the order does not matter. As long as, you get few true for the benchmarks it means that your code is working correctly. The benchmarks that are going to be used for grading will not have reparation in them.
- The size of the of the input files is not constant. You should not make any assumption on the length of the input files.
- The time out for sorting an input with 1 million entries is set to be 10 minutes. Make sure that your program can carry the task under this limit.

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