CS342 Operating Systems – Fall 2018
Project 1: Processes and Threads

Assigned: Oct 08, 2018
Due date: Oct 22, 2018, **23:55** (Moodle)

You will do this project individually. You have to program in C and Linux. You are required to use the following distribution of Linux: **Ubuntu 16.04 – 64 bit**.

Part A: Processes [35 points]
Objective: Practice process creation, developing multi-process applications, file I/O, statistics.

In this part, you will develop a **multi-process** application that will generate a value histogram for the values sitting in a set of input ascii text files, one value per line. Values can be an integers or real numbers. The program will be called **phistogram** and will take the following parameters:

phistogram *minvalue maxvalue bincount N file1 ... fileN outfile*

Here, *minvalue* is the minimum value that exists in the given set of input files, and *maxvalue* is the maximum value. *bincount* is the number of bins in the histogram. Let *w* denote bin-width. Then w = (*maxvalue* - *minvalue*) / *bincount*. The first bin will give the count of values in range [*minvalue*, *minvalue*+*w*); the second bin will give the count of values in range [*minvalue*+*w*, *minvalue*+*2w*); and so on. *N* is the number of input files. *file1 … fileN* are the names of these input files. *outfile* is the output file.

Your program will create another child process for each input file to generate a histogram for the values in that input file. Hence there will be *N* child processes working concurrently on the *N* input files, and at the end, *N* histograms will be generated into *N* intermediate files. The parent process will then combine these *N* histograms into one histogram and will output this histogram to the output file. Each output line will contain information about a separate bin in the following format: *binnumber*: *count*. Binnumbers will start at 1.

Part B: Threads [35 points]
Objective: Practice developing multi-threaded applications.

In this part, you will develop the same application described in part A using threads. For each input file, there will be a separate worker thread. Each worker thread will generate the histogram of the corresponding input file values into a global data structure in memory. Then the main thread will read those histograms from these structures and will generate a single histogram and will print that out to the output file. The program will be called **thistogram**.

Part C: Experiments [30 points]
Objective: Practice designing and conducting experiments and applying knowledge and skills acquired in the Probability and Statistics course.

Do some timing experiments to answer the following questions.

a) What is the running time of your multi-process application for 1, 2, 4, 8 processes working together for the same input? What it the running time of your multi-treaded application?

b) Consider a 2 process (or 2 thread) application. What is the running time for various values of input size?

Submission

Put all your files into a project directory named with your ID, tar the directory (using **tar xvf**), zip it (using **gzip**) and upload it to Moodle. For example, a student with ID 20140013 will create a directory named 20140013, will put the files there, tar and gzip the directory and upload the file. The uploaded file will be 20140013.tar.gz. Include a **README.txt** file as part of your upload. It will have your name and ID at least. Include also a **Makefile** to compile your programs. We want to type just **make** and obtain the executables. Do not forget to put your report (PDF form) into your project directory.

Additional Information and Clarifications

- *Suggestion: work incrementally; step by step; implement something, test it, and when you are sure it is working move on with the next thing.*
- More **clarifications**, additional information and explanations that can be useful for you may be put to the **course website**, just near this project PDF. Check it regularly.
- You can use Piazza for questions and discussions.
- The objective in this project is not to see a speed-up by use of multiple processes or threads. The objective is to practice use of multiple processes and threads.