CS342 Operating Systems - Spring 2016
Project 2: Multi-threaded Programs

```
Assigned: Feb 24, 2016 Wednesday
Due date: Mar 07, 2016 Monday, 23:55
```

You will do this project individually. You can do it on a virtual machine if you wish. You have to program in Linux using C programming language.

Objective: Practicing thread creation, thread data sharing, and multithreaded programming (by use of POSIX threads).

Part A [70 points]

Write the same program of project 1 part A using threads now. You will develop the same program indexgen, but this time you will use threads to do concurrent/parallel processing. No need to use message queues since threads can exchange data using global variables. There will a main thread and several worker threads. The main thread will read words from the input file as in project 1, and will partition the words (together with their line numbers) into lists of items (an item is a structure containing a word and its line number). There will be one list (of items) per worker thread. The number of worker threads, N, will be given as a command line parameter to your program. After generating the list for a worker thread completely, the worker thread will start sorting the items (according to words). Main thread can create a worker thread after the corresponding list has ben generated. Main thread will not apply any sorting (no insertion sort either). After generating the lists for the worker threads, the main thread will wait for worker threads to complete their work. Then, it will take the lists and will write the content to an output file and generate the index in this way. Then the worker thread will terminate as well.

Please read the project 1 part A specification for remembering what indexgen program, the main process, and worker processes were doing. The maximum number of worker threads, MAXTHREADS, is 30. Minimum is 1. You can assume that the max line size (including newline character) is 4096 in an input file. Maximum word length is 64 including the NULL character at the end of a string (word).

The program executable will be named as **t_indexgen** and will have the following parameters:

```
t_indexgen <n> <infile> <outfile>
```

Here <n> is the worker count, <infile> is input text file (ascii file) and <outfile> is output text file. An example invocation can be:

```
t_indexgen  8 in.txt out.txt
```

Part B [30 points]

Solve the programming problem 4.2 in the 9th edition of our textbook: “An interesting way of calculating pi (π) is to use a technique known as Monte Carlo". The problem 4.2 asks you to use a single worker thread that will compute a count of random points in a circle. Here you will use multiple threads: N threads other than the main thread. Each one of the N worker threads will randomly generate K points (as described in the textbook) and will find out the count of points that are within the circle and put the count into a global variable. When the worker threads finish, the main thread will compute the sum of these counts and will use that to estimate the value of pi.

Your program will be called pi and your source file will be called pi.c. It will be run as follows:

```
pi  <N> <K>
```

Here <N> is the number of worker threads (maximum 30), and K is the number of random points that will be generated by each thread. In total KN random points will be generated. The output will be just a single floating point number (of type double) representing pi.

Submission:

Put the following files into a project directory, tar the directory (using **tar xvf**), zip it (using **gzip**) and upload it to Moodle. You will also upload to PAGS (programming assignment grading system) if the teaching assistant requests it.

- t_indexgen.c: contains your C program
- pi.c: contains your C program
- Makefile: Compiles the programs.
- README: Your name and ID and any additional information that you want to put.

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.
- The following web page contains a simple project skeleton. You can clone it into your local machine, if you wish.
    - https://github.com/korpeoglu/cs342-spring2016-p2
- There are references on the website of the course about threads and multi-threaded programming.