CSCI 340 - Operating Systems I

Assignment 5 (Part I of II) Total Points 50

Objectives

In this assignment you will develop a small software application that uses a Monte Carlo simulation to estimate PI. For more information about how this simulation works see youtube video link[1]. This assignment will allow you to gain experience, or extend your knowledge, in the following areas:

- **More 'C' Programming**: This includes variable declaration, data types, arrays, pointers, operators, expressions, selection statements, looping statements, functions, structs, and header files.
- **Threads**: Learn how to create, run, join, and terminate multiple threads.
- **Resource limits**: Learn how to use system calls to get resource limits (such as the maximum number of threads for a single process).
- **Timing**: Learn how calculate the amount of time required for a block of code to execute.

System and Standard Lib Functions

In this assignment you will use the system and standard library functions listed below. Please become familiar with the syntax and usage of these calls. Detailed information about each function listed below can be found using the `man` command from the console: i.e. `man pthread_create`, will show the man page (short for manual page) for the `pthread_create` function.

- **Thread Creation**: `int pthread_create(pthread_t* thread, const pthread_attr_t* attr, void* (*start_routine) (void*), void *arg)`
- **Thread Join**: `int pthread_join(pthread_t thread, void** retval)`
- **Thread Exit**: `void pthread_exit(void* retval)`
- **Resource Limit**: `int getrlimit(int resource, struct rlimit* rlim)`
- **Timing**: `int clock_gettime(clockid_t clk_id, struct timespec* tp);`

Provided Files

The three files listed below are provided to you.

- **mcsim.h**: Header file that defines the Monte Carlo simulation function prototypes used in this assignment. **Please note**: You may not modify or add new function definitions to this header file.
- **mcsim.c**: The file containing the implementation of the functions listed in *mcsim.h*. **Please note**: You may not modify the functions implemented in this file.
- **hw5.c**: Source code file that includes a stubbed out main function to be completed by you. **Please note**: This is the only file you will modify.

[1] https://www.youtube.com/watch?v=VJTFfIqO4TU

Todo

The *hw5.c* is provides step-by-step instructions to be completed by you. Please read the comments carefully and follow their instructions.

Collaboration and Plagiarism

This is an **individual assignment**, i.e. **no collaboration is permitted**. Plagiarism will not be tolerated. Submitted solutions that are very similar (determined by the instructor) will be given a grade of zero. Please do your own work, and everything will be OK.

Submission

Create a compressed tarball, i.e. *tar.gz*, that only contains the completed *hw5.c* file. The name of the compressed tarball must be your last name. For example, *ritchie.tar.gz* would be correct if the original co-developer of UNIX (Dennis Ritchie) submitted the assignment. Only assignments submitted in the correct format will be accepted (no exceptions). Submit the compressed tarball (via OAKS) to the Dropbox setup for this assignment. You may resubmit the compressed tarball as many times as you like, Dropbox will only keep the newest submission.

To be fair to everyone, late assignments will not be accepted. Exceptions will only be made for extenuating circumstances, i.e. death in the family, health related problems, etc. You will be given a week to complete this assignment. Poor time management is not excuse. Please do not email assignment after the due date, it will not be accepted. Please feel free to setup an appointment to discuss the assigned coding problem. I will be more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution. Additionally, code debugging is your job. You may use the debugger (gdb) or print statements to help understand why your solution is not working correctly, your choice.

Grading Rubric

For this assignment the grading rubric is provided in the table shown below.

Program Compiles	10 points
Program Runs with no errors	10 points
main() function implementation	30 points

In particular, the assignment will be graded as follows, if the submitted solution

- does not compile: 0 of 50 points
- compiles but does not run: 10 of 50 points
- compiles and runs with errors: 15 of 50 points
- compiles and runs without errors: 20 of 50 points
- main function correctly implemented: 50 of 50 points