New York University
Tandon School of Engineering
Department of Computer Science and Engineering

Introduction to Operating Systems
Fall 2024

Assignment 7
(10 points)

Write a program that uses a multi-threaded (for speedup) Monte-Carlo simulation to estimate the probability of two students in our class having the same birthday.

Your program's main routine shall create a number of worker threads NUM_THREADS=4 (defined as a macro) in order to speedup the computation (you may name the common routine "`void* WorkerThread(void* pvar)`" and use a shared variable (an integer) as well as synchronization primitives (e.g. a semaphore or mutex).

Each thread shall perform a number of trials NUM_TRIALS=100,000 (defined as a macro), in which it creates a list of n random numbers, passed to your program as a parameter (in our case, test with n=23 and n=49), each has a value between 0 and 364 representing each person's birthday within the year. If (at least) two of them coincide, the thread increments a **shared** variable `nhits` by 1 (for each trial). The shared variable holds the total number of times the experiments/trials succeeded (i.e. 2 or more students had a matching birthday).

Thus the total number of trials is NUM_THEADS*NUM_TRIALS. When all threads complete, we shall calculate and output the probability as:

```
nhits/(NUM_THREADS*NUM_TRIALS)
```

For a class of 49 students, it should be 97%, whereas for 23 students, it should be about 50% (https://en.wikipedia.org/wiki/Birthday_problem). After verifying the correct operation, use your program to compute the probability for classes of size 41 and 128 students.

Some useful notes:

- Use the man pages for more info on how to use a semaphore or a mutex.
- Each thread **must** seed the random number and use its own state so it won't match the other thread's state, and thus each thread **shall** have a different sequence of random numbers, for example:
  ```
  unsigned int rand_state = (unsigned int) time(NULL) + pthread_self();
  ```
- You need to use rand_r() to generate the random numbers and not rand(), for example:
  ```
  rand_r(&rand_state)
  ```
- Note that you will need to use the -pthread option with gcc in order to link the pthread library.

What to submit:

Please submit the following files individually:

1) Source file(s) with appropriate comments.
   The naming should be similar to "**lab#_$.c**" (**#** is replaced with the assignment number and **$** with the question number within the assignment, e.g. lab4_b.c, for lab 4, question c OR lab5_1a for lab 5, question 1a).

2) A single pdf file (for images + report/answers to short-answer questions), named **"lab#.pdf"** (**#** is replaced by the assignment number), containing:
    - Screen shot(**s**) of your terminal window showing the current directory, the command used to compile your program, the command used to run your program and the output of your program.
3) Your Makefile, if any. This is applicable only to kernel modules.

RULES:

- You shall **use kernel version 4.x.x or above**. You shall not use kernel version 3.x.x.
- You may consult with other students about GENERAL concepts or methods but copying code (or code fragments) or algorithms is NOT ALLOWED and is considered cheating (whether copied form other students, the internet or any other source).
- If you are having trouble, please ask your teaching assistant for help.
- You must submit your assignment prior to the deadline.