New York University
Tandon School of Engineering
Department of Electrical & Computer Engineering

Introduction to Operating Systems (CS-GY6233)
Spring 2022

Assignment 5
(15 points)

1) (3 points) Repeat assignment 4 part 3, except that now you shall use a bounded buffer of length BUF_LEN (i.e. when you allocate a shared memory, it should have a size of BUF_LEN for the array). Define that length in a macro (with a value of 10), but make sure I can change that value if I wish to.

```
#define BUF_LEN 10
```

2) (4 points) Repeat assignment 4 part 3, except that now the two processes shall communicate using ordinary pipes.

3) (8 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 4 worker threads 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).

Each thread shall perform a number of trials/experiments 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).

Use 4 threads, thus the total number of trials is 4*NUM_TRIALS. When all threads complete, we shall calculate and output the probability as:

```
nhits/(4*NUM_TRIALS)
```

For a class of 49 students, we expect the number to be about 97%, for 23 students, it should be about 50% (https://en.wikipedia.org/wiki/Birthday_problem).

Some useful notes:

- Each thread may seed the random number and use its own state so it won't match the other thread's state, and thus each thread may have a different sequence of random numbers, for eample:
  ```
  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.

Submission file structure:

Please submit a **single .zip file** named **[Your Netid]_lab#.zip**. It shall have the following structure (replace # with the actual assignment number):

```
└── [Your Netid] hw# (Single folder includes all your submissions)
    ├── lab#_1.c (Source code for problem 1)
    ├── lab#_2a.c (Source code for problem 2a, and so on)
    ├── lab#_1.h (Source code header file, if any)
    ├── Makefile  (makefile used to build your program, if any)
    ├── lab#.pdf (images + Report/answers to short-answer questions)
```

What to hand in (using Brightspace):

- Source files (.c or .h) with appropriate comments.
- Your Makefile if any.
- A .pdf file 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.

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.