Programming project# 2 – PART I (Warmup programming only)
IT383 Spring 2023

All the programs should be compiled and then tested successfully on one of the IT Linux servers before submission.

You may use your own Linux machine running on Virtual Box for development and testing. However,

- Make sure that you frequently back up your programs stored in your own Linux VM on your personal computer!

- Transfer your source code from your local Linux VM to IT Linux server and then compile/run/test your program before submission! Some bugs in your source code might not generate any error/segmentation fault while it is running on your local Linux VM but they might generate some error/segmentation fault on IT Linux servers, which happened often in my previous classes.

Programs containing compilation errors will receive failing grades. Those containing run-time errors will incur a substantial penalty. You should make a serious effort to complete all programs on time. Programming assignments are individual. You should complete them with your own effort.

Part II of Programming project#2 (*actual* programming) will be made available soon.

NOTE: Each program should be placed in its own directory. Also, you need to include a Makefile in the same directory as the source code to allow the instructor to compile your program automatically. Similarly, your half-page write-up in plain text format should be placed in the same directory. You will submit a single "ZIP" file that includes all the subdirectories where your programs and documentation are located. The name of ZIP file should be *yourlastname_firstName_program2part1*.zip

- You might want to use the following Unix command to create a single zip file:

```
% zip –r    yourlastname_firstName_program2part1.zip ./1 ./2
```

NOTE: DO NOT include any *.o or executable files.

Warm-up programming

[1] (25 points) Multithreaded C/C++ program using the Pthread API (Figure 4.9 in textbook)

(1-A) (0 points) **Read Chapter 4.1-4.3:Thread** (if needed, also chapter 3:Process) in **Advanced Linux Programming** (NOTE: A copy of the book can be downloaded from the ReggieNet website or directly from the author's website) carefully and try example code. You need to try the example code in the two chapters to get better understanding of Pthread APIs.

(1-B) (10 points) Create a subdirectory called "1". Change to the subdirectory

Type in/complete/Run/Study the following C programs in **Advanced Linux Programming**.

- Listing 4.1 (*thread-create.c*) Create a Thread
- Listing 4.2 (*thread-create2*) Create Two Threads
- Listing 4.3 Revised *Main* Function for *thread-create2.c*
- Listing 4.4 (*primes.c*) Compute Prime Numbers in a Thread
- Listing 4.5 (*detached.c*) Skeleton Program That Creates a Detached Thread

Explain in detail what you have learned from each sample code. (The total number of words in the document for Listing 4.1-4.5 should be more than 300 words.). Note that a single "Makefile" can be used to compile all 5 files based on timestamps of relevant files.

(1-C) (15 points) Create a subdirectory called "1". Change to the subdirectory

Type in/complete/Run/Study the following C programs in **Advanced Linux Programming**.

- Listing ~~4.12~~ 4.11(job-queue2.c) Job Queue Thread Function, Protected by a Mutex
- Listing 4.12 (job-queue3.c) Job Queue Controlled by a Semaphore
- Listing 4.14 (condvar.c) Control a Thread Using a Condition Variable
-

  Explain in detail what you have learned from each sample code. (The total number of words in the document for Listing ~~4.1-4.5~~ 4.11, 4.12, and 4.14 should be more than 300 words.). Note that a single "Makefile" can be used to compile all 3 files based on timestamps of relevant files.

Deliverables: source code, Makefile, written document (in plain text format)
DO NOT include any *.o or executable files.

NOTE: There are NO main() function in each of the source code for 4.11,4.12, and 4.14.

When you create a makefile for those programs, please use the following approach:

```
all: prog411.o prog412.o prog414.o


prog411.o: prog411.c

	gcc -c prog411.c
```

similarly for 412 and 414 programs.

"gcc" with "-c" option will only compile the given source code and ends up generate ".o" file.