CECS 326 Assignment 4

Due: May 3, 2018

As you have come to understand, the shmp1.cpp and shmc1.cpp you compiled and ran in Assignment 3 have serious deficiency due to race condition. In this assignment you are to correct the problem using the semaphore mechanisms that Linux provides. Note that any corrections you make should not include the removal or changes of the sleep() calls in the shmc1 program.

Two implementations of semaphore are commonly available on most distributions of UNIX and Linux operating systems. The System V implementation includes **semget()**, **semctl()**, and **semop()**, and a **struct** **sembuf** used in the **semop()** calls, which are defined in <**sys/sem.h**>. The POSIX implementation includes **sem\_wait()**, **sem\_post()**, and **sem\_open()**, **sem\_close()** & **sem\_unlink()** for named semaphore and **sem\_init()** & **sem\_destroy()** for unnamed semaphore, which are defined in <**semaphore.h**>. You may use either one of these implementations. Details on the definition of these functions and their use may be found on Linux man pages.

The program must run successfully on a Linux machine in the CECS Lab.

Do the following for this assignment:

1. Add necessary synchronization code in *shmp1.cpp* and *shmc1.cpp* to correct problems due to race condition, and compile them into executables *shmp1* and *shmc1*, respectively.
2. Run your corrected version of *shmp1* (with *shmc1*) to make sure that the output is correct.
3. Submit on BeachBoard and a hard copy of the corrected *shmp1.cpp* and *shmc1.cpp*, with a cover page that provides your name, your student ID, course # and section, assignment #, due date, submission date, a description of these programs with your corrections in them highlighted.
4. The programs must be properly formatted and adequately commented to enhance readability and understanding.
5. Demonstrate your program on a Linux machine in the Lab and explain details of your implementation.