Operating Systems and Kernel Design

Prof. D. B. Megherbi

EECE 4811/EECE 5811

Fall 2023

Assignment #2

This second assignment's focus is on threads and their synchronization. You will make use of the compilation option lpthreads to implement your assignment. There is a need to start your assignment this week. There are three parts to this assignment, which is due 3 weeks from today. See the course schedule.

Part-1: you are provided with 4 fully runnable sample programs. You need to compile and run every one of them and briefly explain in your report what you observe when running test_sem3 and test_4 programs. Explain the difference and why.

Part-2: You are asked to implement a simple main program with **1** (one) thread that we call "*provider*" and **260** (two-hundreds and sixty) threads that we call "*buyers*". Design and implement a program that will handle the synchronization of the *provider* threads and the *buyers'* threads. The program should ensure that the *provider* (thread) continuously and repeatedly produces different integers (here representing items), prints them on the screen, and writes them in a buffer of size x (x<10) that you select. Your program also ensures only one *buyer*(thread) at a time buys an item (represented by an integer) from the same buffer and prints it on the screen. The other *buyers* (threads) should sleep for some time while the thread *buyer* is buying. When a thread wakes up, it tries to *buy* an integer from the buffer. Make sure that each and all *buyers* get a chance to buy an item represented by a number. No *buyer* should make the other ones starve. The *providers* should not stop providing items (i.e., numbers representing items). Make sure that the *provider* gets a chance to insert items (i.e., numbers) in the empty buffer locations where the buyers buy items from non-empty buffer locations. Make sure that each buyer can buy an item from a non-empty buffer location. Only one thread from either the provider thread or the 260 buyer threads can access a buffer location at a time.

Your program should have one command-line argument which specifies the number N of *buyer* threads. When the program starts, it creates N+1 threads which place themselves in a thread pool. The main program issues the initial notification to the *provider* thread to start providing integers and then notifies the main program to start doing its job as described above by making sure that the N *buyer* threads have a chance to buy items (represented by numbers), each buyer thread should at least buy one item.

Part-3: After keeping a copy of program-2 from part 2 above (of course, that will be needed in the report for part 2), modify the program of part 2 and

(a) run it in the case of *one single provider* and *6 buyers*. Explain in your report what one significant change to your code will be to make it efficient, if any.

(b) run it in the case of two (2) *provider* threads and 12 *buyers*.