Alexandria University
Faculty of Engineering
Comp. & Comm. Engineering
CC373: Operating Systems
Fall 2019

جامعة الاسكندرية
كلية الهندسة
برنامج هندسة الحاسب والاتصالات
مادة نظم التشغيل
خريف ٢٠١٩

Lab3
Semaphores

Objectives:

1. Familiarizing with concurrent programming.
2. Handling races, synchronization and deadlock conditions.

Problem Statement:

You are required to write a C program to solve the following synchronization problem using POSIX and "semaphore.h" libraries.

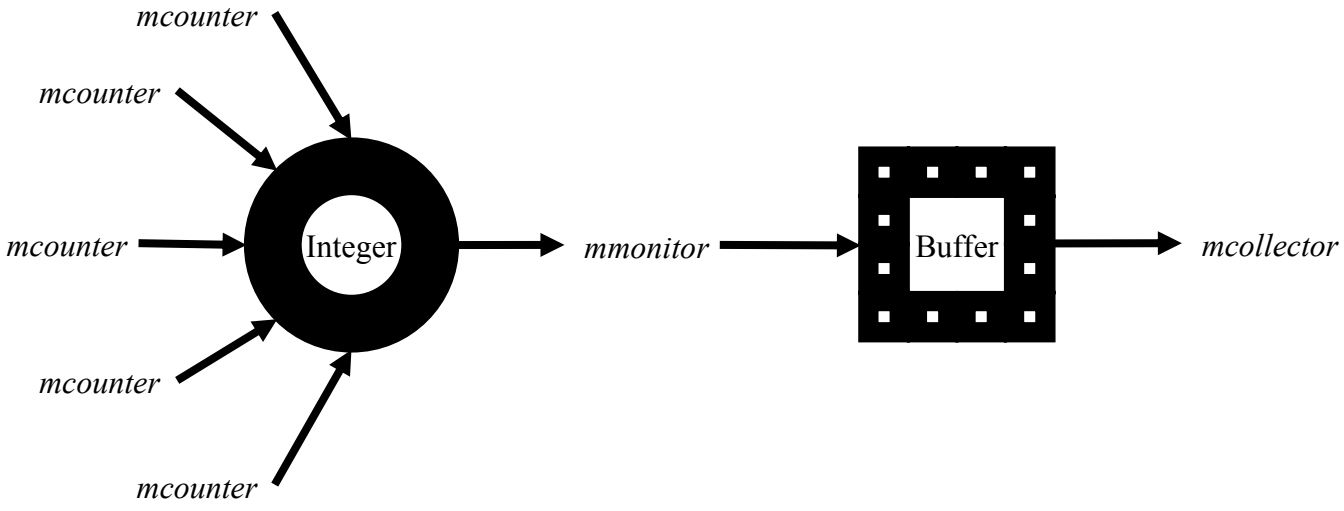
N *mcounter* threads count independent incoming messages in a system, and another thread *mmonitor* gets the count of threads at time intervals of size t1, and then resets the counter to 0. The mmonitor then places this value in a buffer of size b, and a *mmcollector* thread reads the values from the buffer.

Any thread will have to wait if the counter is being locked by any other thread. Also, the mmonitor and mcollecttor threads will not be able to access the buffer at the same time or to add another entry if the buffer is full.

Assume that the messages come randomly to the system, this can be realized if the mcounter threads sleep for random times, and their activation (sleep time ends) corresponds to an email arrival. Similarly, the mmonitor and mcollector will be activated at random time intervals.

Hints:

1. Divide up the problem into two sub-problems
   (a) P1: counter threads count messages and add them to the counter shared variable, and monitor thread reads it.
   (b) P2: monitor thread places the count in the buffer and collector thread reads it.



Sample Run:

1. Your sample run will show a sequence of the behavior of each thread at the times of their activation (at random intervals), for example

Counter thread 1: received a message

Counter thread 1: waiting to write

Counter thread 2: now adding to counter, counter value=??

Collector thread: nothing is in the buffer!

Monitor thread: waiting to read counter

Monitor thread: reading a count value of ??

Monitor thread: writing to buffer at position ??

Monitor thread: reading a count value of ??

Monitor thread: writing to buffer at position ??

Counter thread 3: received a message

Counter thread 1: now adding to counter, counter value=??

Monitor thread: Buffer full!!

Collector thread: reading from buffer at position ??

Materials:

1. Reference functions and samples: http://www.csc.villanova.edu/~mdamian/threads/posixthreadslong.html
2. A solved producer-consumer problem for a buffer of size 1: http://www.cs.arizona.edu/~greg/mpdbook/programs/pc.sems.c

Deliverables:

- Complete source code in ONE FILE, commented thoroughly and clearly.
- Name your file as your ID (e.g., 5237.c, 5237.cpp, 5237.C, 5237.cc, …)
- Submission to the eLearning system. No other means of submission. File submitted to eLearning is the one that will be graded.

Notes:

1. Language used: C/C++, Operating System: Linux.
2. You should work **individually**.
3. You may talk together on the algorithms or functions being used, but are allowed to look at **anybody**'s code.
4. *Revise the academic integrity note found on the class web page.*