Computer Science 330
Operating Systems
Siena College
Fall 2022

Programming Project 2: Producers/Consumers with Semaphores

Due: 11:59 PM, Friday, October 28, 2022

In this small programming project you will write a program to simulate a bounded buffer problem with multiple producers and multiple consumers.

You may work alone or with a partner or two on this programming project.

Learning goals:

1. To work with POSIX threads and POSIX semaphores in a solution to a generalized bounded buffer problem.

Getting Set Up

In Canvas, you will find link to follow to set up your GitHub repository, which will be named `semaphores-proj-yourgitname`, for this programming project. Only one member of the group should follow the link to set up the repository on GitHub, then others should request a link to be granted write access.

Bounded Buffer with Semaphores

In class, we looked at this pseudocode for the bounded buffer problem for a single producer and a single consumer using semaphores for synchronization and mutual exclusion.

- Shared data:

```
semaphore fullslots, emptyslots, mutex;
fullslots=0; emptyslots=n; mutex=1;
```

- Producer process:

```
while (1) {
  produce item;
  wait(emptyslots);
  wait(mutex);
  add item to buffer;
  signal(mutex);
  signal(fullslots);
}
```

- Consumer process:

  ```
  while (1) {
    wait(fullslots);
    wait(mutex);
    remove item from buffer;
    signal(mutex);
    signal(emptyslots);
    consume item;
  }
  ```

Your task is to write a C program that implements the bounded buffer problem for an *arbitrary number* of producers and consumers and items to be processed (each specified by command-line parameters). If i items are to be processed by p producers and c consumers, each producer should produce $\frac{i}{p}$ items and each consumer should consume $\frac{i}{c}$ items. You may either check that these divide evenly and report an error if not, or account for the uneven division by having some producers or consumers process one extra item.

Use POSIX threads to create your producers and consumers and use POSIX semaphores for synchronization. Your solution should avoid the busy wait seen in some of the class examples (other than any busy waiting that might takes place inside a semaphore `wait` or `signal` operation, which is not your fault). Note that the class pseudocode examples, including the one above, were usually for a single producer and a single consumer, so the `in` and `out` variables could be modified by only one process. Once we introduce multiple producers and consumers, we also introduce new critical sections related to concurrent access of those variables, which will also require protection with semaphores.

Write your program in a file called `prodcons.c`. You may (and should!) use the source code from any of the class examples as your starting point and/or to guide your solution. Be sure to indicate clearly which code you borrow and which code you add or modify.

See the examples and man pages for more information on POSIX threads and POSIX semaphores. Please don't hesitate to ask questions!

Submission

Commit and push!

Grading

This assignment will be graded out of 25 points.

Feature	Value	Score
Arbitrary numbers of producers/consumers/items	5	
Correct and efficient usage of POSIX semaphores	15	
Documentation	4	
Working `Makefile`	1	
Total	25	