91.308 Operating Systems   Assn #5: Prep for Donut Factory

Out: Wed Oct 19
Due: Wed Oct 26

You must submit **hardcopy** of the **course cover-sheet**, your **source code**, your **output** and a **write-up** of your results described below.

In this assignment, you will build essential data structures and testing framework required to implement a generalized **producer-consumer** system.

STEP 1: Unix memory segment shared between producer and consumer processes. In this portion of the assignment, you will develop code that demonstrates a Unix shared memory segment. The shared memory segment will hold a single 32-bit integer.

a. Create a simplified **producer process** must create a **Unix shared memory segment** that holds a 4-byte integer. After creating the memory segment, the producer should run an infinite loop which continually increments the integer's value, with a 0.1 second delay in between each increment. When you run the producer in its own Unix shell, it should print out the value of the integer each time it increments it.

b. Now, you must create a simplified **consumer process** that attaches to the same shared memory segment created by the producer. The consumer should be in a loop, continually printing out the value of the integer in the memory segment. Run the consumer and producer simultaneously, and demonstrate that both have access to the same 4-byte integer memory object (i.e, the producer writes the integer and the consumer reads it).

c. Launch a second copy of the consumer, and demonstrate that it too has read access to the shared integer.

(Note that the shared integer is just a sample case to demonstrate that any object can be shared in memory across processes. In the next assignment, the shared memory segment will hold 4 racks of donuts, represented as integer arrays.)

Turn in for Step 1: *A code listing of your 'producer' and 'consumer' processes, and a screen snapshot that shows the 3 Unix shells running simultaneously (1 producer and 2 consumers).*

STEP 2: Defining semaphores and implementing mutually-exclusive access to the shared integer. In this portion of the assignment, you will create Unix semaphores, and use them to provide protected, mutually-exclusive access to the shared integer object.

a. Modify your producer process to create and initialize the semaphore.

b. In the producer, wrap the operation of incrementing the integer with down and up calls on the semaphore. Include print statements inside the protected region along the lines of "producer has control of the semaphore… producer is incrementing the integer… producer is releasing the semaphore."

c. Modify the consumer to also increment the integer. Add code so that it can use the same semaphore to create protected access to this region, just like the producer. Include print statements.

d. Run the code in two terminal shells, demonstrating that the producer and consumer alternate their access to the integer.

Turn in for Step 2: *A code listing of the new producer and consumer processes, and a screen snapshot that shows the 2 Unix shells running.*

Other notes:

- The course **cover-sheet** should include a statement as to how much success you felt you had with this project, and your **write-up** must provide a **discussion of your results and any observations or problems you encountered in the project.**

- Make sure you include the paths to your code and any build details (makefile) on the **cover-sheet.**

- You will find a sample code to help you with the assignment on our dept file system at:

```
~fredm/308/donut/
```

**Credits: This assignment was developed by Prof. Moloney, UML CS Dept.
Prof. Martin has revised it by breaking it down into two parts,
with intermediate deliverables between the two.**