COSC 625: Programming Project #2: Producers and Consumers

Distributed: 29 September 2010

Due: 20 October 2010 (Excess time given because of overlap with Arduino project -- this project as a stand alone should take about a week of frustration)

You may do this project in groups of AT MOST TWO. I encourage you to work alone on this project after you are able to get the IN CLASS simple producer/consumer working correctly.

Synopsis:
Use Java or C/C++, and only semaphores and mutexes for coordination and protection -- do not use `synchronizing` or other useful Java aids to coordination and communication. Write code with multiple producers and multiple consumers all accessing the same (only one) data variable that is referred to as the mailbox. The mailbox is able to contain a single `int`.

There are three types of tasks: main, producer, and consumer. Each producer task is an instantiation of a Producer extension to Thread. Each consumer task is an instantiation of a Consumer extension to Thread.

The main will first output "task *main* initiating", then will create all tasks and other "real-time objects" (e.g., semaphores). All producer tasks will share the same run code for producers. All producer tasks start by outputting, "producer task *id* initiating". All consumer tasks will share the same run code for consumers. Similarly, all consumer tasks start by outputting, "consumer task *id* initiating". Each task has a unique id that can be used to control the path of execution through the run code.

Every producer will create a data item with same value as the producer's id. The producer will output "producer task *id* creates *datum*". The producer will post the item to the mailbox. The producer will then go to sleep for a random amount of time between 1 and 5 seconds. After 5 iterations, producers will create a poison pill and post it to the mailbox, then output the string "producer task *id* terminating" and terminate.

Every consumer will obtain a data item, output it to the console with "consumer task *id* acquires *datum*". If the value is not the poison pill, the consumer task will increment a local counter and continue. If the value is a poison pill, the consumer will ***write the poison pill back to the mailbox***, output "consumer task *id* terminating" and terminate.

Only main() will be modified between the various configurations (i.e., different runs with different numbers of producers and consumers). The code for the producer and consumer tasks may not be modified.

Deliverables:
- Demonstration with the following configurations:

- One producer, one consumer
- Two producers, one consumer
- Two producers, two consumers
- Five producers, two consumers

Only `main`() will be modified between the various configurations. The code for the producer and consumer tasks may not be modified.

Be prepared to change to different configurations during the demo.

- Code walk-through

- Hard copy of code

Grade based on:

Satisfying specifications	65%
Satisfying style requirements	25%
Code elegance	10%

In a group, both members must be able to demonstrate and explain. *A member who is not able to independently explain and modify the code, will have the grade reduced by 30%.*