91.308 Operating Systems Assn #6: The Donut Factory (aka, the producer-consumer problem)

Out: Fri Oct 28
Due: Mon Nov 7

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

The problem you must solve has been described in class and is formalized follows:

1. The problem is a **producer-consumer** problem requiring a **single producer** (of donuts) process, and an **arbitrary number of** consumer processes. The producer must: **create a shared memory segment** with 4 ring buffers and their required control variables, one for each of four kinds of donuts

2. You must **create semaphores** for each ring buffer to provide mutually exclusive access to each buffer after creating and mapping the segment.

3. The producer will enter an endless loop of calling a **random number** between **0 and 3** and producing one donut corresponding to the random number (remember that the producer could get blocked if it produces a donut for a ring buffer which is currently full). A donut can be thought of as an integer value placed in the ring buffer, where the integer is the sequence number of that type of donut (i.e. the initial entries in each ring buffer would be the integers 1 to n, where n is the size of the buffer, and when the 1st donut is removed by a consumer, the nth+1 donut can be placed in the buffer by the producer)

4. Consumers are started **after the producer** (any number may be started at any time after the producer) and must find the **shared memory ID** created by the producer and **attach the segment** to their images. Each consumer must:

    a. **get and attach the shared memory segment,** which contains the ring buffers and control variables

    b. **get and use the semaphores** created by the producer to coordinate access to each donut type

    c. **enter a loop** of some number of **dozens of iterations** and begin **collecting dozens of mixed donuts** using a **random number** as does the producer to identify each donut selected

    d. each time a consumer **completes a selection of a dozen donuts**, the process makes an entry in a **local file** (use your own naming convention) as follows:

```
process PID: 34567    time: 10:22:36.344     dozen #: 4

plain    jelly    choc-creme    honey-dip
11         3       9              15
38         7       20
           11      24
           19      30
           24
```

5. Your output for submission should include a catenation of each of consumers' local files, **in the order that you started your consumers**

6. 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 **quantitative discussion of your results and any observations or problems you encountered in the project.** For this problem, it is your write-up which will determine your grade. Writing code to implement the assignment is step one, but experimenting with different configurations and providing your results and conclusions in your write-up is most important. Make sure you include the paths to your code and any build details (makefile) on the **cover-sheet.**

7. You must run a test set with a **producer** and **5 consumers**, where each consumer runs until it collects **10 dozen donuts** using a queue size of **50 slots**, and submit these complete results (since the probability of deadlock in such a configuration is known to be low, you should be able to get results here with one or two tries… you'll need a full run to generate all the consumers' output files). Once you've generated results for this fixed configuration, you must experiment with the **queue depth setting** in additional runs (however many more you think you need) to collect enough data to allow you to **produce a graph** of the **probability of deadlock vs. the depth of the queues** for the 5 consumer, one producer case. Is this distribution **linear or non-linear**? Summarize and discuss these results, your initial results, and any other observations you've made in your write-up (don't include the consumer output files for these runs, just summarize what you found and include your graph).

8. You will find a help file to assist you in using the system calls at:
   `http://www.cs.uml.edu/~fredm/308/donut/donut.c`
   `http://www.cs.uml.edu/~fredm/308/donut/utility.h`

   Please note that these files cannot be directly compiled! Instead, you should extract pieces of the code and use them in your own "producer.c" and "consumer.c" programs.

Credits: This assignment was developed by Prof. Moloney, UML CS Dept.