COMP 3080 Operating Systems Proj #3 September 27, 2016

1. The **due date** for this assignment is **Thursday, October 20**.

2. **All** of your submissions must include a minimum of **four** separate files:
   - **File 1:** A short **write-up** that first specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0.
   - **File(s) 2(a, b, c, …)**: Your **complete source code**, in one or more **.c** and/or **.h** files
   - **File 3**: A **make file** to build your assignment. This file must be named **Makefile**.
   - **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you annotate it so that it is self descriptive and that all detailed output is well identified.

3. The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your submit command (see the on-line file **Assignment_Submit_Details.pdf** for specific directions).

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

   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

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

   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)

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:

get and attach the shared memory segment which contains the ring buffers and control variables

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

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

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      coconut    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 your consumers' local files, **in the order that you started your consumers**

6. Your write-up 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.

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. From step 7, you must find the **50% deadlock queue size** and use this specific queue size to generate a **second deadlock probability distribution** that varies the number of consumers **from 1 to 10** using the fixed queue size previously determined. Prepare a **graph for these results** and discuss these results in your write-up (don't include the consumer output files for these runs, just summarize what you found and include your graph).

9. You will find a system call help file to assist you in using the system calls at the URL: **http://www.cs.uml.edu/~bill/cs308/call_help_assign3.txt** and you will also find a shell script to help you collect data at the URL: **http://www.cs.uml.edu/~bill/cs308/A3_donut_loop.sh**