91.515 Operating Systems Proj #2 September 22, 2011

1. This assignment is due on **Thursday, October 13.**

2. You must submit electronically as described below, a **write-up** describing your degree of success, your methods of testing and the **graphs** discussed in class, along with your **source code**, and your **output listings** as described below (see iterm #4)**.** If you have done this project before (as part of my 91.308 undergraduate course) you must redo the assignment on a Windows system using the Win32 system call API. See the URL:

http://www.cs.uml.edu/~bill/cs515/Assign_Submit_Readme.pdf

3. 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 process solution** using threads within a process (the pthread interfaces are suggested and available on all of our Linux systems). Unlike the first assignment, **multiple producers** (of donuts) must be created as threads along with an arbitrary number of consumer threads. The main routine of the single process should setup and initialize the required ring buffers and control variables in global memory where they can be seen by all other threads of the process. The main routine will then begin creating some number of producer threads and some number of consumer threads.

The **producer threads** must:

- find the global structure consisting of 4 ring buffers and their required control variables, one for each of four kinds of donuts

- find the necessary locks for each ring buffer to provide non-conflicting access to each buffer the producers will enter an endless loop of calling a random number between 0 and 3 calling a random number between 0 and 3 and producing one donut corresponding to the random number (remember that the producers could get blocked if they produce 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- 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 a producer)

Consumers are started **after the producers** (any number may be started at any time after the producers) and must find the global buffers, control variables and locks and then:

- 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 thread should make an entry in a thread-specific local file (use your own naming convention) as we did in assignment #1, and as we've previously described, a thread should perform an operation like a usleep() to force a context switch to another thread so the donuts are distributed somewhat uniformly among threads:

Consumer thread #: **3** time: **10:22:36.344** dozen #: **4**

plain	jelly	coconut	honey-dip
11	3	9	15
38	7	20	
	11	24	
	19	30	
	24		

4. You must run a test set with **30 producers and 50** consumers, where each consumer runs until it collects 200 dozen donuts, and submit a listing of the first 10 dozen donuts obtained by any 5 of the 50 consumers. You must determine the 50% percent deadlock queue size for the given configuration as we did in assignment #1, showing a distribution of DL probability based on queue size. You must also run **at least 4 other configurations**, with 100, 150, 250 and 300 dozens collected and summarize and discuss these results, along with your initial results, in your write-up, graphing the probability of DL distributed by dozens collected for the 50% queue size determined in the first experiments. **Your write-up is the most important part of this assignment**, and should provide a detailed discussion (and probability graphs as in assignment # 1) of the deadlock frequencies found as a function of queue size, and donuts collected.