COMP 3080 Operating Systems Proj #4 March 4, 2025

1. This assignment is due on **Thursday, March 27.**

2. **All** of your submissions must include a minimum of **four** separate files:
   - **File 1:** A **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. The **write-up** for this project is the **most important part of the project**, and must include the **graphs** discussed in class and detailed below. **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. We have separate submit directions for the 9:30 class and for the 12:30 class, so **make sure** that you consult the correct submit file for your section:
Assignment_Submit_Details_9_30.pdf for the 9:30 section 201
Assignment_Submit_Details_12_30.pdf for the 12:30 section 202.

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 process solution** using threads within a process (the pthread interfaces available on all of our linux systems, as well as most other POSIX based systems). Unlike the first assignment, **multiple producers** (of donuts) must be created as threads along with a multiple number of consumer threads. The main routine (initial thread IT) 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 consumer threads and some number of producer threads.

The **producer threads** must:

- begin using the global structures consisting of 4 ring buffers and their required control variables, one for each of four kinds of donuts (each ring buffer requires an in pointer, an out pointer, a space counter and a donut counter and we must also have a serial counter for each flavor of donut)

- 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 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 of that flavor can be placed in the buffer back at slot 0 by a producer)

The **consumer threads** 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 random numbers as does a producer to identify each donut selected

- each time a consumer thread 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. In assignment #1, the output files were just standard-out redirected by the script into a specific file such as c1, c2, etc.. We can't do that here, so each consumer (or at least 5 of the 50 in a given run) must create and write to a named file, but only 10 dozen records should be written by a consumer before the file is closed, even though the consumer will be consuming 2000 dozen in a given run. Each consumer (or each of the 5 specific consumers selected) should write its first 10 records (dozen format shown below) to its file. 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. The dozen format you must use is shown here:

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		

You must run a test set with **30 producers and 50** consumers, where each consumer runs until it collects **2000 dozen donuts**, and submit a listing of the first 10 dozen donuts obtained by any 5 of the 50 consumers (this composite file or individual files must be well labeled). You must determine the 50% percent deadlock queue size (for this configuration you should start with a queue size in the 1500 – 2000 slot range) for the given configuration as we did in assignment #1, graphing a distribution of DL probability based on queue size (five points in the graph should be fine). Holding the 50% queue size determined above fixed, you must now run **at least 4 other configurations**, with **1000, 1500, 2500 and 3000 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. **Your write-up is the most important part of this assignment**, and should provide a detailed discussion (and probability graphs as in assignment # 3) of the deadlock frequencies found as a function of queue size, and donuts collected.

You can find assignment slides and help code at:

http://www.cs.uml.edu/~bill/cs308/call_help_assign_4_slides.pdf
and
http://www.cs.uml.edu/~bill/cs308/call_help_assign4.txt