COMP 5150 Operating Systems 1
Project #3 Due Date December 9
October 14, 2021

- This assignment's due date is **Thursday, December 9**.

  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 details on page 2) :

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

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

  1. The problem is a **producer-consumer** problem requiring an arbitrary number of producers (of donuts), an arbitrary number of consumers and a buffer manager. In addition, a node controller process is required on each host which could ever have any producer or consumer processes running on it. (Producers and consumers may actually be implemented as threads or processes as you choose, but the **buffer manager** must be a **process** which creates either child processes or threads to handle each request which arrives from a producer or a consumer.)

  2. The **buffer manager** must:

  A. provide and control storage for **4 ring buffers** and their required control variables, one for each of four kinds of donuts

  B. provide buffer under-run and over-run protection to each buffer while providing optimal concurrency (producers and consumers active at the same time). This requirement will ensure that a producer-consumer interaction leading to corruption of the buffers can never occur, but consumer-consumer corruption (or producer - producer corruption) **is not controlled in any way by the buffer manager**

  C. after initial set up the buffer manager will enter an **endless loop**, waiting for connection requests from a consumer or a producer and handling each request in a child process (you may use dynamically or statically created threads instead of child processes if you choose). There is **no checking** by the parent buffer manager on the number of children active at any time, and if more than one consumer child (or more than one producer child) for the same donut flavor is active at the same time it could lead to corruption of the buffers. **It is the responsibility of the node controllers to assure that this will never happen**

  D. the buffer manager will have to be terminated by a special signal when all consumers have finished. **If the buffer manager creates child processes, it should also have a signal manager to clean up the children when they become zombies.** The node location of the buffer manager is arbitrary, but only a single buffer manager process instance can exist at any time

3. The **producers** must:

   A. obtain a random number between **0 and 3** to determine what donut flavor to produce and then connect to the **local node controller** to make a request for permission to produce that flavor donut. When the node controller **grants permission**, the producer must get the **next donut number** of the required flavor from a **serial counter that is shared by all producers on the local node**, and then **increment** that serial counter. The producer must now establish a connection to the buffer manager and send along its produce request, including the donut number and its node ID (remember that a producer could get blocked in buffer manager code if it produces a donut for a ring buffer which is currently full). The buffer manager must take the request and form an **element** to insert into the appropriate ring buffer. An **element** can be thought of as a structure of two integers placed in a ring buffer slot, where the integers include the node ID of the producer and the local serial number of that type of donut (i.e., the entries in each ring buffer slot would include a unique producer node ID along with a unique donut number for that flavor from that node). The buffer manager must now return success to the producer and close the connection.
   B. after connection, a producer must **inform his node controller** of his completion, and begin the produce process again (forever);

4. **Consumers** are started after the producers (any number may be started at any time after the producers and on any number of nodes in the network). They must:

   A. connect to the node controller on their node and begin calling a random number between 0 and 3, preparing to request one donut corresponding to the random number (remember that the consumer could get blocked in buffer manager code if it requests a donut from a ring buffer which is currently empty).
   B. after connection a consumer must then negotiate with his node controller for permission to send his request to the buffer manager for retrieval of the requested type of donut. **When permission is obtained, the consumer must connect to the buffer manager**, send the request, receive the donut and disconnect and inform the local node controller of his completion
   C. consumers loop through some number of dozens of iterations and begin collecting dozens of mixed donuts using a random number as in previous assignments. Producers should produce forever, leading to being blocked in their code (as in previous assignments) when the consumers finish, and the buffer manager's queues get filled. (You must have s strategy for killing these blocked producers and the buffer manager when the problem is done or if the problem becomes deadlocked.)
   D. **each time a consumer completes a selection of a dozen donuts, it makes an entry in a local file** (use your own naming convention, but consumer files must be uniquely named across the network) as follows (this example uses the two integer numbering scheme described above) :

process name and node ID time dozen #

ID plain	**ID jelly**	**ID coconut**	**ID honey-dip**
01 11	03 3	01 9	02 19
02 3	01 7	01 20	
02 11	03 24		
01 19	01 30		
01 24			

5. **Node controllers** must be started on every node where a producer or consumer may run (at least 3 hosts). We have a set of machines available for testing and running your code (they allow open use of the network ports) and can be reached using an ssh to cs.uml.edu on port 308. Please see the URLS:

   http://www.cs.uml.edu/~bill/cs515/A3_NET_MACHINES_README.txt
   and
   http://www.cs.uml.edu/~bill/cs515/A3_NET_MACHINES_README.txt

   The node controllers that you run must:

   A. each have knowledge of the existence and location of each other node controller, and be able to establish a **lifetime connection** to each other controller (this will require some initial configuration information to be available at execution time to all controllers)
   B. each have a **unique node ID** which they and all other controllers know about
   C. each allow an arbitrary number of local producers and/or consumers to establish a connection to them for the purpose of synchronization control
   D. each enforce **Lamport's (or Ricart's) mutual exclusion algorithm** across all nodes, participating with producers and consumers at their site to assure that producer - producer (or consumer - consumer) corruption cannot occur

- Your submission should include **your write-up, your source code, any make files to build your targets, and the output listings described below**. I would like you to place all these objects in a directory and then submit just the directory as discussed in class.

- Your brief write-up should include **the required graphs** described below, any performance information you obtained, and any observations or problems you encountered in the project.

- You must run a test set with **5 producers, a buffer manager and 15 consumers** divided up with at least one producer and at least one consumer on each of at least three different hosts

- Each consumer must run until it collects **10 dozen donuts**. You must then select a successful run and provide a separate listing for any **5 of your 15 consumers**. These five listings will comprise the output portion of your submission.

- As with previous assignments, you must find the queue size that leads to **50% deadlock for the above configuration**, and then collect data for that queue size

with 3, 10, 20 and 30 consumers for the same 5 producers, and summarize and discuss these results, along with your initial results, in your write-up. You should include a graph for the **initial configuration showing DL probability** as a function of queue size, and a **second graph based on varying the consumer count as described above** (fixed queue size, variable consumers).