**Producer and Consumer**

Benjamin Carter and Trevor Pope

Grand Canyon University

CST-315: Operating Systems Lecture & Lab

Professor Ricardo Citro

January 24th, 2024

# CLC - Assignment 1: Producer and Consumer

**Names and Roles**

Trevor Pope – Wrote the producer/consumer logic and wrote the testing/validation script. Also designed the flowchart.

Benjamin Carter – Wrote the threading/main loop code and the implementation segment in the report. Also wrote comments in the source code.

**GitHub Link:** [**https://github.com/BenRobotics101/CST-315**](https://github.com/BenRobotics101/CST-315)

**Implementation**

In a Producer and Consumer model, a queue is used to store data that is created by the producer which is later fed into the consumer. In this model, a consumer receives data as long as the queue contains data. The consumer receives data in a FIFO method. A producer can input data as long as the queue is not full.

This model is implemented through a static queue data structure. One variable stores the beginning index of the queue and the second variable holds the length of the queue. This system allows for “wrapping around” for full efficiency of the static array. When adding a value to the queue, the end pointer (calculated by (start + length) modulus size) increases. When a value is consumed from the queue, the start pointer increments by one with the result modulus size. This modulus operator allows that “wrapping around” for efficient data management. This also is quick, without requiring if-statements or branching.

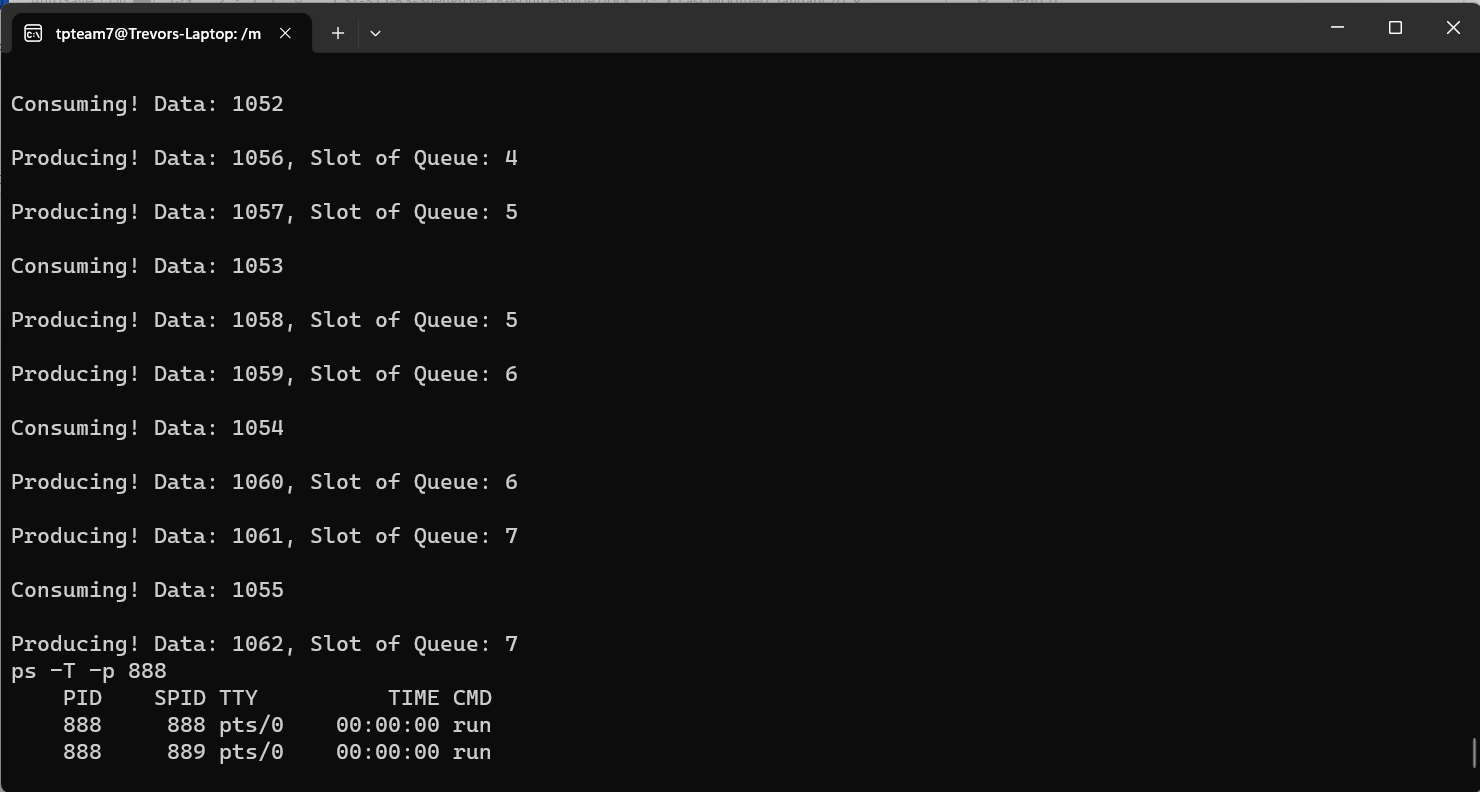
A queue can be tested to see if it is full if the length equals the size of the array. A queue is empty if the length equals zero.

A thread is created which acts as the producer, producing data four times a second. The original flow of execution becomes the consumer. It consumes data also four times a second after a short delay.

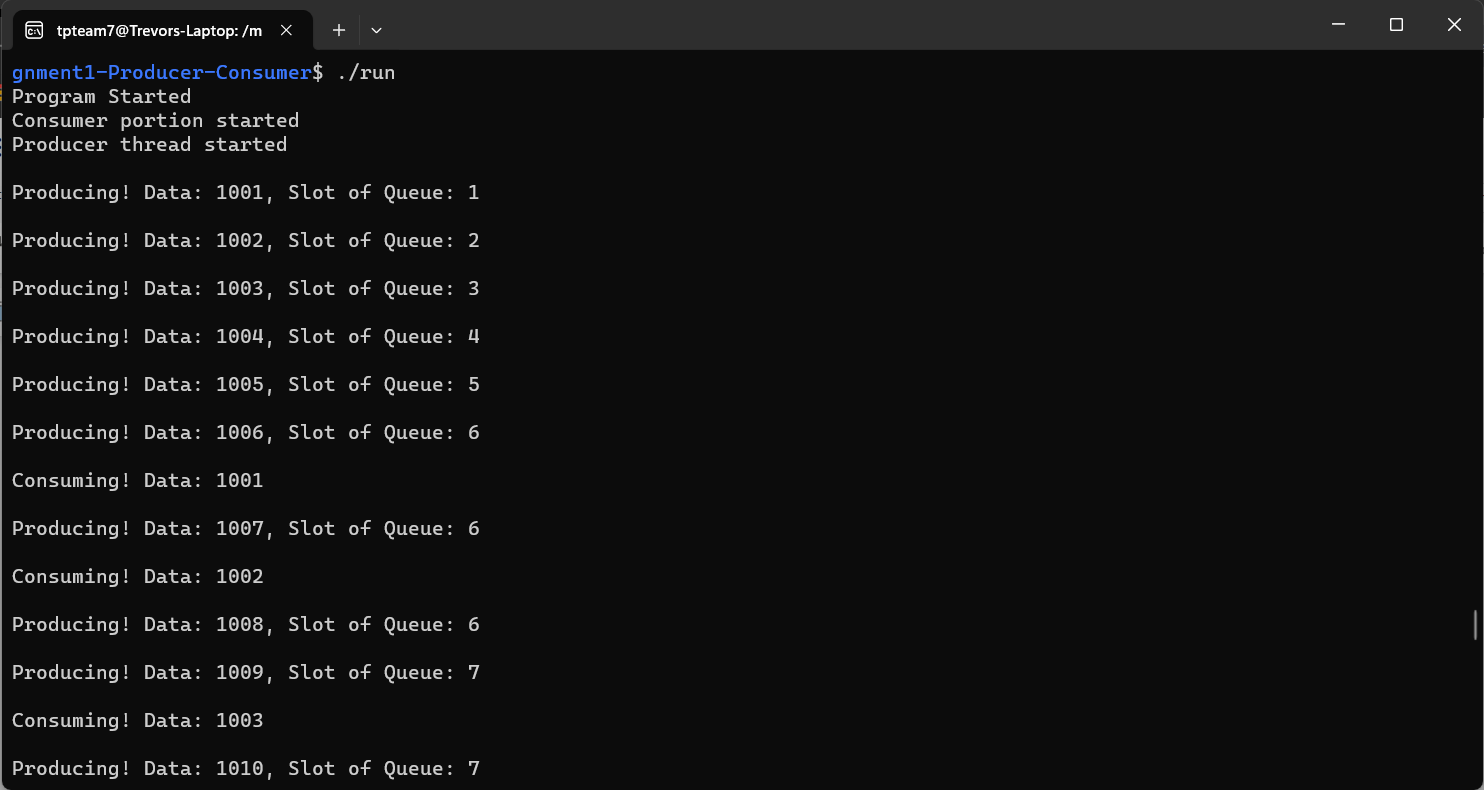
**Testing and Validation**

When implementing the queue, there were zero bugs seen. Our process operated correctly as it used two separate threads, one for the consumer and one for the producer. The program then generated data using the producer thread which inputs data into the queue as long as the fixed queue length is not reached. At the same time, the consumer thread is running which will get data from the queue if it is not empty. We don’t have any examples of the program consuming when the queue is empty or producing when the queue is full. To validate that the program was operating correctly, we checked to make sure that two threads were running for the process using the ps –T –p <pid> command in the shell. We saw that the two threads set up to run on the program were executing. This could also be seen in the execution of our program as both the consumer and the producer are running, which they were set up to run exclusively on their specific thread.

Here is a screenshot that shows the process running with the threads:



**Screenshots**



**Flowchart:**