Joseph Pennington

EECS 687 Lab08 Report

1. We are solving the producer and consumer problem. This problem occurs when a consumer and a producer share a common buffer for communication. The producer is creating data and puts it in a common buffer that is then consumed by a consumer. There can be multiple produces and consumers sharing the same common buffer. The problem arises when either the producer is filling the queue to quickly or the consumer tries to empty an empty queue. The solution ensures that the producer and consumer threads run concurrently in sync and resolves any synchronization problems that arise from race conditions. The solution will also block a consumer when it tries to consume data from an empty buffer, and the consumer should be resumed when the buffer is not empty. On the other hand, the solution will also block a producer when it tries to put data in a full buffer, and it will be resumed when the buffer is not full. In general, the producers and consumers pass data to each other via the common buffer.

2. The busy-wait solution, while correct, is inefficient for threads running on the same CPU because it is wasting many cycles by forcing each thread to spin when the conditions required for them to continue are not met. These resources should be used by other threads that are adding or removing data from the buffer.

3. I am confident that my solution is correct because in each test case, the 30 items are both produced and consumed. They are also only consumed after they are produced and only consumed once. If the queue is full, the producer should print that the queue is full and wait until it is no longer full before producing a new item. Similarly, if the queue is empty, the consumer will print that the queue is empty and wait until the queue is no longer empty before checking if there is data to consume. Each producer and consumer exit appropriately because once the maximum number of items is produced, the producers will print their exit statements. Similarly, the consumers will exit once the maximum number of consumed data points is reached. My solution passes the tests with different numbers of producers and consumers because each producer and consumer acts in the appropriate manner and there are no deadlocks, infinite loops, or busy-waiting scenarios.