Program Description

This program presents the Producer–Consumer problem using threads, semaphores, and mutual exclusion. It uses two processes, a producer and a consumer, that share the same table (the buffer), and they must operate in synchronization.

The producer generates items and places them into the table. The consumer takes (or "consumes") items from the table. If the table/buffer becomes full, the producer must wait until the consumer removes an item. If the table is empty, the consumer must wait until the producer adds an item.

Explanation of Key Components

1) Shared Memory Use

The shared memory in this program is implemented through a shared buffer, which both the producer and consumer processes access concurrently. This buffer allows produced items to be stored temporarily until the consumer is ready to consume them.

2) Semaphores

Semaphores are a synchronization tool that provides more sophisticated ways to process synchronize their activities. In this program, semaphores are used to control when the producer and consumer can access the shared buffer.

3) Mutual Exclusion

Mutual exclusion is used to make sure that only one thread or process accesses a shared resource at any time. In this program, it is used to make sure the producer and consumer are not accessing the shared buffer at the same time.

Executed Results

When the program runs, both the producer and consumer execute concurrently.

The picture below shows the executed results of the Producer-Consumer program: