Producer-Consumer Problem

Brody Peelman

Dr. Qiang Guan

30, October 2024

Operating Systems

Implementation

Producer process initializes mutex lock semaphore for ensuring Mutual Exclusion throughout the Producer-Consumer executions. Consumer reads in semaphore. Both processes open the shared memory table. Producer requests lock first to generate starting item to the table. In execution, both processes can generate or remove either one or two items at a time respectively. This variation is implemented through a random number generation and proceeds accordingly.

Execution is observed as a process requests the mutex lock. If the other is not in critical section, the mutex lock will be available and the process requesting is granted a turn. This process will either generate or remove an item from the shared memory table and release the lock. The other process is granted the lock upon request. The scenario provided in code details eight generations and removals of items of the shared table. At the end of each execution, there is cleanup. The shared memory table is closed and unlinked and so is the mutex lock semaphore.

The shared memory table is implemented using the “sys/mman.h” library. The mutex lock semaphore is implemented using the “semaphore.h” library. Several sleeps are embedded into the files to reflect proper waiting between the processes. These sleeps are implemented using the “unistd.h” library.

Example Output