Producer/Consumer Processes

# Summary

The producer process and consumer process utilize semaphores to synchronize their producer and consumer threads. Each thread is associated with a specific spot in the table and will either produce or consume only that value. They both utilize the shared.hpp file in order to impose the structure of the table on the mmap(). There can only be one of each of these processes running to ensure proper execution of them and proper closing of memory.

## producer.cpp

This must be run in order for the consumer process to begin its consumer threads. It will produce the first two values without regard for the consumer process. It waits for the consumer process to consume the value and then continues. Each thread produces fifty values. This is just an arbitrarily chosen amount that could be more, less, infinite, or dependent on some I/O.

## consumer.cpp

This consumes the data by printing out what it has consumed to the console. Any operation could be done here but for demonstration of proper mutual exclusion I just output to console. This process also handles the cleanup of the shared memory portion as it consumes data without regard for whether or not the producer process has exited and without regard for the amount of data. As such flags are used, along with an additional semaphore, to know when to cleanup the shared memory.

## shared.hpp

This is a simple file that contains a struct to define the shared memory portions layout for the processes to mmap() them properly. It holds semaphores, flag variables for consumer threads, and the two variables on the table.