Producer Consumer Problem

Name: Shail Rakhunde

Roll No. BT10CSE065

The Producer Consumer Problem (also known as bounded buffer problem) is popularly used to illustrate the multi-process synchronization condition when many users/processes try to access a fixed size buffer to process a stream of data. The problem divides the users/processes into two categories: Producers and Consumers.

Producers produce data after some process and put it into the shared buffer. The consumers on the other hand, take data out of the buffer and process it. The shared buffer in between producers and consumers is usually implemented with FIFO (First In First Out) data structure such as Queue. Stack sometimes may also be used for buffer with some specific purpose.

To illustrate Producer Consumer Problem in a more real life way, we can consider the Printing Operation. The printing basically involves two processes done by two different systems. First a computer or any other host system creates, formats and finalises data for printing, and when ready puts this data into the buffer shared with printer. The printer when ready starts taking out the data sequentially from buffer and prints it on paper. The problem comes when the buffer is of limited size which may be is less than the size of data to be printed or the two systems are operating at different speeds.

This gives rise to the following cases:

1. Producer system is faster and fills up the buffer, while consumer system is not able to keep up.
2. Consumer system is faster and empties the buffer, while producer is not adding data at that rate.
3. Multiple producer/consumer threads are producing/consuming data. A condition may occur when two producer thread find a single space in buffer and try to use it, or two consumer thread find a single data item in buffer and try to consume it.

These are the conditions for data transfer between any two multi-processing systems acting as Producer and Consumer. And hence it is quite a general problem faced by every system programmer in system development.

Solution

To implement and simulate the producer consumer problem, I am using threads in java to create and depict multiple users/processes scenario. By default 3 threads of each type i.e. producer and consumer is created. We can have more/less threads by defining/removing the thread objects of corresponding type.

The total size of data producer is going to produce is currently set to 50.

Thread\_0, 1, 2 represents producer threads in the program, while Thread\_3, 4, 5 represents consumer threads.

The data produced by producer and put into buffer is integer. And the places where multiple threads operating same integers creates synchronization problem, Atomic Integers are used. Atomicintegers provide basic integer facilities such as increment-decrement and ensures the thread-safe operations and hence are mainly used as counters in multi-threaded applications.

To ensure mutual exclusion of different threads on same buffer, java keyword ‘synchronized’ is used. The synchronized block allows only a single thread to use the buffer during a sequence of instructions, and prevents other threads from executing this sequence if some thread is currently executing it.

The use of threads eliminates the 3rd condition of Producer Consumer Problem.

To eliminate remaining conditions, the thread is made to sleep whenever its operation is not desired. That is, when the buffer becomes full, the producer threads are put to wait and if the buffer becomes empty, the consumer threads are put to wait. And whenever a data is put into buffer or taken out of buffer, all the threads are notified to operate.

This sums up the Producer Consumer Problem.

Thank you for reading.