OPERATING SYSTEMS – PRODUCER CONSUMER PROBLEM

DECEMBER 15, 2016

16CSCI03I - OPERATION SYSTEMS

By Mohamed El Sayed

Operating Systems – Producer Consumer Problem

Formal problem definition

In computing, the **producer–consumer problem**[1][2] (also known as the **bounded-buffer problem**) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time. **The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.**

The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer. The solution can be reached by means of inter-process communication, typically using semaphores. An inadequate solution could result in a deadlock where both processes are waiting to be awakened. The problem can also be generalized to have multiple producers and consumers.

Short problem definition

The producer consumer problem is a “classic example of multi-process synchronization problem” which means given any normal programming problem, you should be able to notice whether or not this problem falls below the producer-consumer type of problems.

How to identify the producer consumer type of problems?

Simply, you will notice that the producer-consumer model has the following items:

1. At least two processes / threads involved.
2. Shared resource (fixed-size buffer / array / file / any form of data).
3. Communication between the N processes is **mandatory** so the program behaves in a valid manner.
4. The solution relies on inter-process communication and there will appear a clear need to use Semaphore.

Explicit producer consumer problem example

Use semaphores to solve the producer consumer problem. Assume the existence of a single reader, single writer, a character array buffer, and two semaphores; one labeled empty, the other labeled full. Other variables may be instantiated as needed.

Solution: https://goo.gl/qvaiwh

Implicit producer consumer problem example:

PS: What does implicit mean? It means that the question did not mention anything which has “producer consumer” and left it for the student to figure out.

Implement the bounded buffers problem using the POSIX thread API (pthreads). There is thread(s) writes a sequence of natural numbers in ascending order, starting with 1. The numbers are written into a bounded buffer. Another set of thread(s) read out the natural numbers and verify that the numbers are in correct ascending order. (This should help you to detect synchronization problems.)

The bounded buffer should be described by a single data structure which contains all information shared between the threads. For example, the declaration

```
typedef struct buffer {
    unsigned int    count;
    unsigned int    data[BUFFER_SIZE];
    int             in;
    int             out;
} buffer_t;
```

Introduces a structure `buffer_t` which contains the buffer (data), the number of element filled with data (count), the buffer index where the next number can be inserted (in) and finally the buffer index where the next number can be retrieved (out). For a correctly synchronized solution, you will have to add elements to this structure as needed. A minimal solution for this problem supports one writer and one reader thread. You are encouraged to write an enhanced solution which supports a runtime configurable number of writers and readers threads. It is possible to get up to 2 bonus points for such an enhanced solution. Note: To compile and link threaded programs, you should pass the -D REENTRANT flag to the compiler and the -lpthread flag to the linker.

Solution (optional to study): https://goo.gl/0AWsOc

More resources you should use in case needed: https://goo.gl/WLRkb7