[BOUNDED-BUFFER PROBLEM]

[Documentation]

DECEMBER 19, 2022
DR. AHMED HESHAM
Eng. Islam Gamal

✞ Solution Pseudocode: -

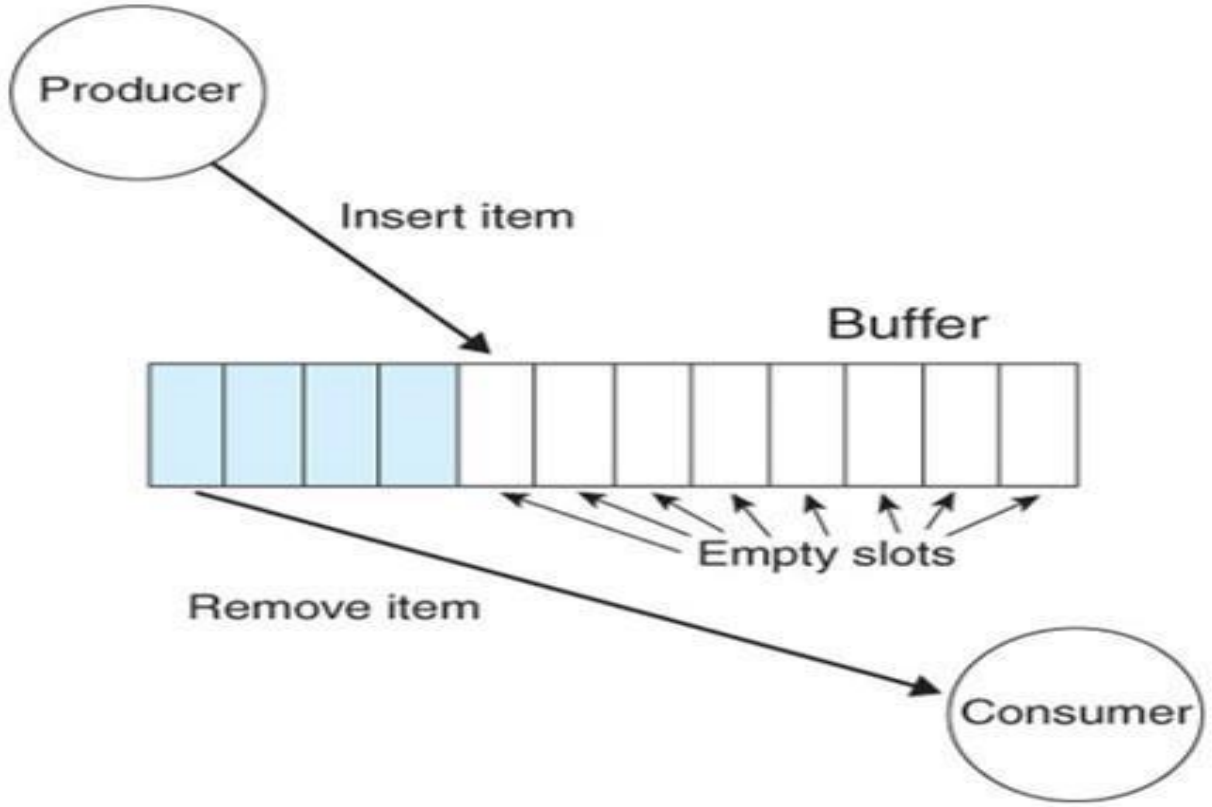

Fig (1): problem overview.

- ❾ Buffer[n] buffer; //initially empty.
- ❾ Semaphore empty; // initially equal n.
- ❾ Semaphore full; // initially equal zero.
- ❾ Semaphore mutex; // initially equal 1.

1. The Pseudocode for the Producer function…

```
do
{
 // wait until empty>0 and then decrement 'empty'
wait(empty);   wait(mutex); // acquire lock
 /* perform the insert operation in a slot */
```

```
 signal(mutex); // release lock
// increment 'full'
signal(full);
}
 while(TRUE)
```

****

2. The Pseudocode for the Consumer function…

```
    do
{
// wait until full>0 and then decrement 'full'
wait(full);
 wait(mutex);  // acquire the lock
/* perform the remove operation
  in a slot */
 signal(mutex); // release the lock
// increment 'empty'
signal(empty);
} while(TRUE);
```

✞ Deadlock: -

What is deadlock?!

- A deadlock occurs when two or more threads wait forever for a lock or resource held by another of the threads.

- A deadlock in OS is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process. The four necessary conditions for a deadlock situation to occur are mutual exclusion, hold and wait, no preemption and circular set. We can prevent a deadlock by preventing any one of these conditions. There are different ways to detect and recover a system from deadlock.

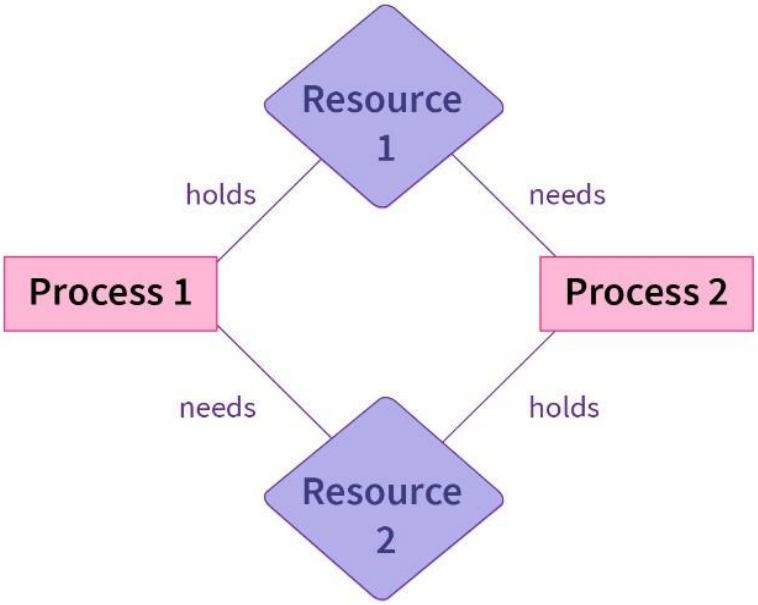

Fig (2): Deadlock overview.

What is deadlock in Bounded-Buffer?!

- The Bounded-Buffer is a Producer Consumer Queue Where The Get method blocks if the buffer is empty and the Append method blocks if

the buffer has reached its maximum allowed capacity and while the code looks correct it contains a nasty deadlock bug.

How to solve deadlock in Bounded-Buffer?!

By using 3 Semaphore:

- Semaphore empty; // initially equal n.
- Semaphore full; // initially equal zero.
- Semaphore mutex; // initially equal 1.

For mange the Two Process (Append – Get) Which Lead to Deadlock Situation.

For example:

1. If empty==0 ----> Producer.block();
2. If full==0 ----> Consumer.block();

✞ Starvation: -

What is Starvation?!

- A process that is present in the ready state and has low priority keeps waiting for the CPU allocation because some other process

with higher priority comes with due respect time. Higher-priority processes can prevent a low-priority process from getting the CPU.

- threads are also waiting for each other. But here waiting time is not infinite after some interval of time, waiting thread always gets the resources whatever is required to execute thread run() method.

- Starvation describes a situation where a thread is unable to gain regular access to shared resources and is unable to make progress. This happens when shared resources are made unavailable for long periods by "greedy" threads. For example, suppose an object provides a synchronized method that often takes a long time to return. If one thread invokes this method frequently, other threads that also need frequent synchronized access to the same object will often be blocked.

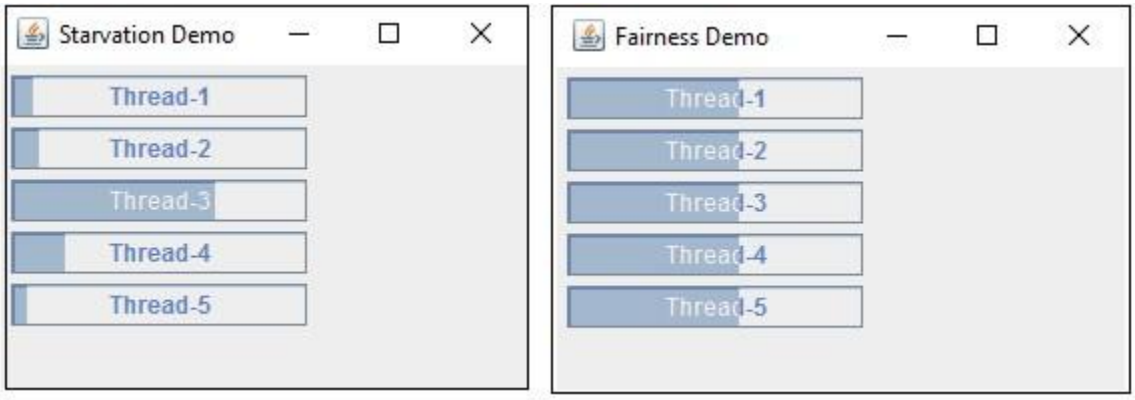

Fig (3): Starvation overview.

What is Starvation in Bounded-Buffer:

- Starvation occurs in Producer Method when Multiple Thread Want to Put item in the buffer and no place in buffer to put any item, them will block themselves in entry-Set Without order of Arrival time and when place is empty, they will compute with themselves to Put the item in buffer and only one of Threads win this fight.

- Starvation occurs in Consumer Method when Multiple Thread Want to Get item from the buffer and no item in buffer to get (Empty-Buffer), them will block themselves in entry-Set Without order of Arrival time and when any place is full, they will compute with themselves to Get the item from the buffer and only one of Threads win this fight.

- How to Solve Starvation in Bounded-Buffer:
  - For this we will use Semaphore with Fairness Constructor.

    Semaphore s = new Semaphore(1,true).
  - By this Semaphore We will Built Semaphore with Priority Queue to block (producer – consumers) in order of arrival time.

✞ Explanation For Real World: -

- Our Real World Is Shop for Trading in Phones…

1. Buffer : is the store Where We Stored Our Items (Phones).

2. Producer : This is the factory we deal with that provides us with phones .

3. Consumer : These are the people who come to buy phones from us

Producer :

When the factory that provides us with phones will come with goods to us here, they will start testing the store, is there a place for it or not…

1. If store (buffer) is full ❼ He will wait.
2. If store (buffer) has empty places ❼ he will put phones in the buffer.

Consumer :

When the customer who is going to buy a phone, he will also do a test on the store. ..

1. If store (buffer) is empty ❼ He will wait.
2. If store (buffer) has full places ❼ he will get phones from the buffer.