Bounded Buffer Problem

Sufyan Ahmad
201980013

Classical Synchronization

- ❏ Why we use
- ❏ Problems in Synchronization

Why we Use Classical Synchronization?

Classical synchronization ensures correct and predictable behavior in concurrent programs by preventing data corruption, race conditions, and maintaining data integrity and order among threads or processes. It involves techniques like locks and semaphores to coordinate access to shared resources and enable thread communication.

Problems in Synchronization:

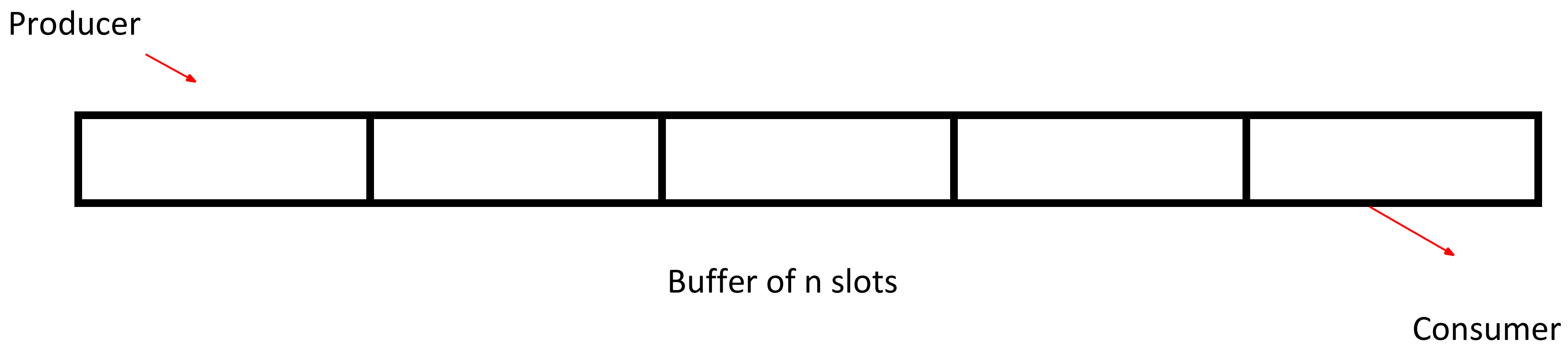
- ❖ Bounded Buffer Problem
- ❖ Reader Writers Problem
- ❖ Dining Philosophy

Bounded Buffer Problem

Also known as Producer Consumer Problem.

There is a buffer of n slots and each slot is capable of storing data.

There are two processes running name as producer and consumer. Which are operating on the buffer.



Ensure and Conditions:

Ensure:

A. Producer tries to insert data into an empty slot.
B. Consumer tries to get data from a filled slot.

Conditions:

1. Producer must not insert data when the buffer is full.
2. Consumer must not remove data when the buffer is empty.
3. Producer and Consumer must not insert and remove data Simultaneously.

Solution of Bounded Buffer problem using Semaphores

We will make use of three semaphores.

- ❖ m(mutex), a binary semaphore which is used to acquire and release the lock.
- ❖ empty, a counting semaphore whose initial value is the number of slots in the buffer, since, initially all slots are empty.
- ❖ full, a counting semaphore whose initial value is 0.

Producer

```
do{

wait (empty);     // wait until empty>0
                   and then decrement 'empty'

wait (mutex);     // acquire lock

 //add data to buffer

signal (mutex);   //release lock

signal (full);     // increment full

}
while(TRUE)
```

Consumer

```
do {

wait (full);   //wait until full>0 and
                then decrement full

wait (mutex);   // acquire lock

//remove data from buffer

signal (mutex);     //release lock

signal (empty);     //increment 'empty'

}
while(TRUE)
```

Signal and Wait:

Definition of Wait()	**Definition of Signal()**

P (Semaphore S) {
 	
    while (S<= 0); //no operation
 	
S - -;
}
	V (Semaphore S) {
	
	    S++;
	}

Producer Consumer Problem:

Mov ax,5

Producer	Consumer
Mov dx,ax //dx=5	Mov bx,ax //bx=5
Inc dx //dx=6	Dec bx //bx=4
Mov ax,dx //ax=6	Mov ax,bx //ax=4

Simultaneously Worked

THANK YOU