Synchronization

Page No.
Date

★ Synchronization or process synchronization is the task of co-ordinating the execution of the process in such a way that no two processes can have access to same shared data and resources.

- categorized as one of the following two types [on the basis of synchronization]

(i) Independent Process → does not affect execution of other process

(ii) Co-operative Process → affect the execution of other process

★ Race condition is a situation where multiple processes access and manipulate the same data concurrently and outcome of the execution depends on the order in which the instructions execute.

★ Critical section is a code segment that can be accessed by only one process at a time.

- Critical section contains shared variables which need to be synchronized to maintain consistency of the data variables.

```
do
{
    Entry section
        critical section
    Exit section
        remainder section
}
while (True);
```

(i) Entry section:- Each process needs permiss to enter critical section, each process goes to critical section through entry sec
(ii) Exit section:- Each critical section is terminated by exit section.
(iii) Remainder section :- The code after exit section is remainder section.

★ Solution to Critical Section must satisfy following three ~~condition~~ requirement.

(i) Mutual exclusion:- If process is executing in critical ~~region~~ section ~~and other~~ the no other process is allowed to execute in the critical section.

(ii) Progress :- In no process is executing in critical section and other process is waiting outside the critical region, then only thoese process that are not executing in remainder section can participate in deciding which will enter next in critical section.

(iii) Bounded waiting :- A bound must exists on number of times a processes are allowed to enter critical section, after a process has made a request to enter critical section.

Page No.
Date

★ • Semaphore is a variable used to control access to a common resource by multiple processes and avoid critical section problem.
• It is an integer variable, which can be accessed only through two operations wait() and signal().
• Advantages
(i) allows Only one process into critical section
(ii) There is no resource wastage.
• Types of semaphores
(i) Binary semaphore
(ii) Counting semaphore.

★ Bounded-Buffer Problem. also known as
• Producer consumer problem is a classical synchronizational problem.
• We can solve this by using semaphores.
• by creating two counting semaphores "full" and "empty" to keep track of the current number of full and empty buffers respectively.
• Producers produce a product and consumers consumes the product, but both use one of the containers each time.
• Producer:- (i) Creates item and adds to buffer
(ii) Do not want to overflow buffer.
• consumer:- (i) Removes items from buffer.
(ii) Do not want to get ahead of producer.

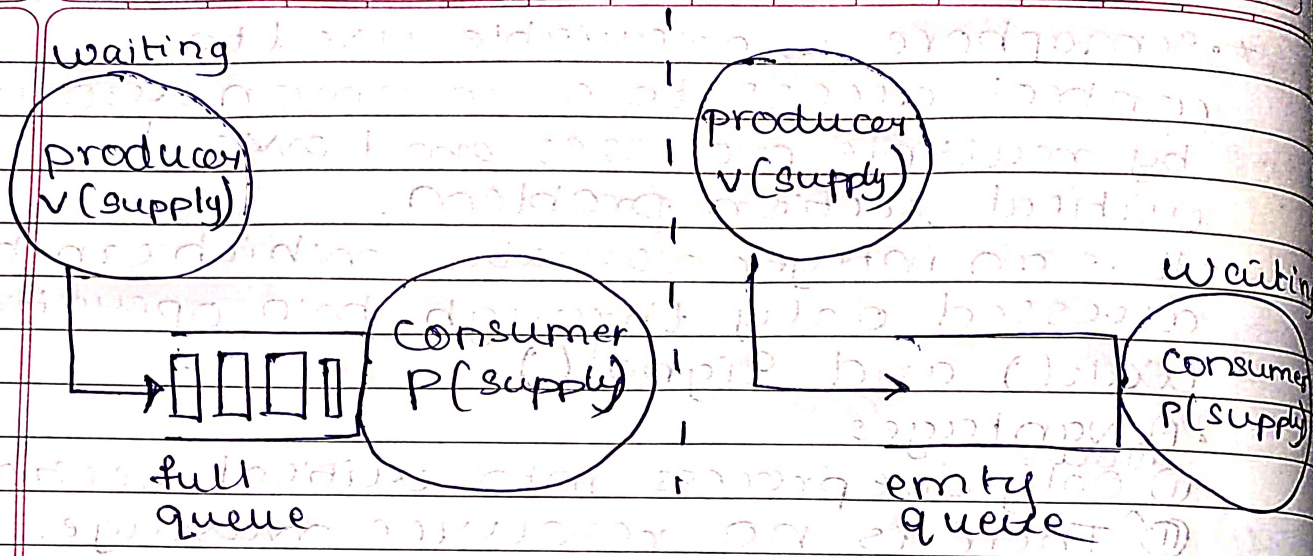

fig:- Producer & Consumer.