**IMPLEMENTATION OF IPC USING SHARED MEMORY**

Interprocess Communication or IPC provides a mechanism to exchange data and information across multiple processes, which might be on single or multiple computers connected by a network.If a longer job has been assigned to the CPU then many shorter jobs after it will have to wait. This algorithm is used in most batch operating systems.

**CHARACTERISTICS** : Characteristics Of Inter-process Communication:

There are mainly five characteristics of inter-process communication in a distributed environment/system.

Synchronous System Calls: In the synchronous system calls both sender and receiver use blocking system calls to transmit the data which means the sender will wait until the acknowledgment is received from the receiver and receiver waits until the message arrives.

Asynchronous System Calls: In the asynchronous system calls, both sender and receiver use non-blocking system calls to transmit the data which means the sender doesn’t wait for the receiver to acknowledge.

Integrity: Messages must be reliable, If the messages are guaranteed to be delivered without being lost is called validity. arrive without corruption and duplication to the destination.

Validity: Point to point message services are defined a

Ordering: It is the process of delivering messages to the receiver in a particular order. Some applications require messages to be delivered in the sender order i.e the order in which they were transmitted by the sender.

**SYSTEM CALLS USED** :

**ftok**(): is used to generate a unique key.

**shmget**(): int shmget(key\_t,size\_tsize,int shmflg); upon successful completion, shmget() returns an identifier for the shared memory segment.

**shmat**(): Before you can use a shared memory segment, you have to attach yourself to it using shmat(). void \*shmat(int shmid ,void \*shmaddr ,int shmflg).

**shmdt**(): When you’re done with the shared memory segment, your program should detach itself from it using shmdt(). int shmdt(void \*shmaddr);

**shmctl**(): when you detach from shared memory,it is not destroyed. So, to destroy shmctl() is used. shmctl(int shmid,IPC\_RMID,NULL);

**IMPLEMENTATION** :

We know that to communicate between two or more processes, we use shared memory but before using the shared memory what needs to be done with the system calls, let us see this −

* Create the shared memory segment or use an already created shared memory segment (shmget())
* Attach the process to the already created shared memory segment (shmat())
* Detach the process from the already attached shared memory segment (shmdt())
* Control operations on the shared memory segment (shmctl())