Preemptive Scheduling

The act of moving a 'Running' process/task into the 'Ready' queue by the scheduler, without request from the process is known as 'Preemption'. Preemptive scheduling is used to implement preemptive multitasking model. Following are the various types of preemptive scheduling algorithms:

- Preemptive SJF Scheduling/Shortest Remaining Time (SRT)
- Round Robin (RR) Scheduling
- Priority Based Scheduling

Key features of preemptive scheduling algorithms

1. Preemptive SJF Scheduling/Shortest Remaining Time (SRT)
   - Sorts the 'Ready' queue when a new process enters the 'Ready' queue
   - Checks whether the execution time of the new process is shorter than the remaining of the total estimated time for the currently executing process.
   - If the execution time of the new process is less, the currently executing process is preempted and the new process is scheduled for execution.

2. Round Robin (RR) Scheduling
   - Each process in the 'Ready' queue is executed for a pre-defined time slot in the order of its arrival in the 'Ready' queue
   - If the process completes before the pre-defined time slice, the next process in the 'Ready' queue is selected for execution
   - When all the process in the 'Ready' queue is executed for the pre-defined time period, the scheduler picks the first process in the 'Ready' queue again for execution.

3. Priority Based Scheduling
   - Priority is assigned to each task
   - When a high priority process is entered in the 'Ready' queue, it is immediately scheduled for execution (without waiting for completing the current task as in non preemptive scheduling)

TASK COMMUNICATION

In a multitasking system, a task may interact with other task. The mechanism through which processes/tasks communicate each other is known as Inter Process/Task Communication (IPC). Based on the degree of interaction, processes can be classified as:

- Co-operating Processes
- Competing Processes

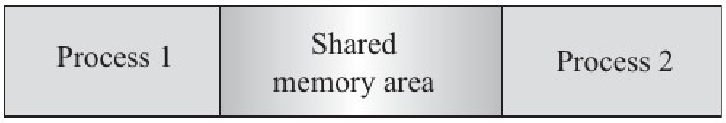
In the co-operating interaction model, one process requires the inputs from other processes to complete its execution. In the competing processes model, processes do not share anything, but shares system resources such as files, display device etc.

Following are some important IPC mechanisms:

- Shared Memory
- Message Passing
- Remote Procedure Call (RPC) and Sockets

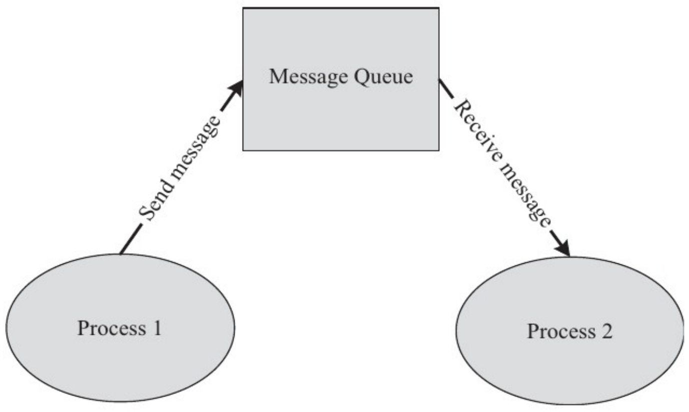
Shared Memory:

In this method, processes share some area of the memory to communicate among them. Information to be communicated by the process is written to the shared memory area. Other processes which require this information can read the same from the shared memory area.



Message Passing:

In this method, when a process wants to talk with other process, it posts a message, which is stored in a Message queue. From the queue, it passes to the desired process.



In certain Real-Time Operating Systems, instead of Message queue, a Mailbox is used. It is usually used for one way messaging.

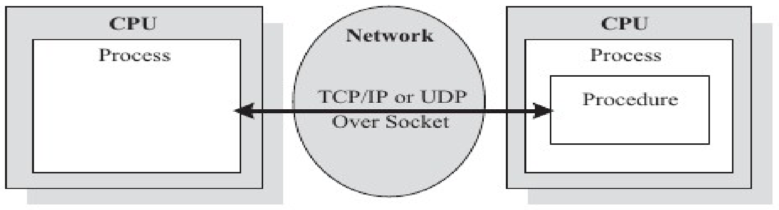
The functions of Message queue and Mailbox are same but Message queue supports more messages.

Comparison of Shared memory and Message passing:

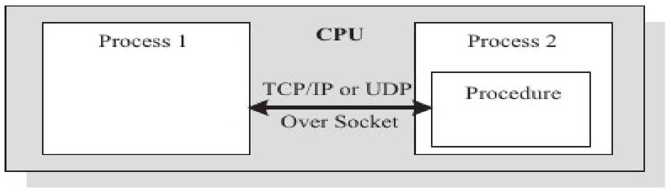
Shared memory	Message Passing
Supports more data	Support limited data
Slow operation	Relatively fast
Has synchronisation overhead	No synchronisation overhead
Implemented using: Pipes, Memory mapped objects	Implemented using: Message queue, Mailbox

Remote Procedure Call (RPC) and Sockets:

In this method, a process can call a procedure of another process running on the same CPU or on a different CPU which is interconnected in a network. (In object oriented terminology, it is called Remote Method Invocation – RMI). RPC is mainly used for distributed applications. RPC can be used to communicate over a heterogeneous network (i.e. network where client and server applications are running on different operating systems).



Processes running on different CPUs which are networked



Processes running on same CPU