3. Round Robin Scheduling Algorithm

- In this algorithm, before allocating the CPU for processes, some **fixed time slice or time quantum** gets decided in advanced, and at any given time control of the CPU may remains allocated with any process maximum for that decided time-slice, once the given time slice is finished of that process, it gets suspended and control of the CPU will be allocated to the next process again for maximum that decided time slice and so on..., each process gets control of the CPU in a round robin manner i.e. cpu gets shared among processes equally.
- If any process completes its execution before allocated time slice then control of the CPU will be released by that process and CPU gets allocated to the next process as soon as it is completed for effective utilization of the CPU.
- There is **no starvation in RR Scheduling algorithm**.
- This algorithm is **purely preemptive**.
- This algorithm **ensures minimum response time.**
- If time slice is minimum then there will be extra overhead onto the CPU due to frequent context-switch.

4. Priority Scheduling

- In this algorithm, process which is having highest priority gets control of the CPU first, each process is having priority in its PCB.
- priority for a process can be decided by two ways:

1. **internally** – priority for process can be decided by an OS depends on no. of resources required for it.
2. **externally** – priority for process can be decided by the user depends on requirement.

- Minimum priority value indicates highest priority.
- This algorithm is purely preemptive.
- Due to the very low priority process may gets blocked into the ready queue and control of the CPU will never gets allocated for such a process, this situation is referred as a starvation or indefinite blocking.
- **Ageing:** it is a technique in which, an OS gradually increments priority of blocked process, i.e. priority of blocked process gets incremented after some fixed time interval by an OS, so that priority of blocked process becomes sufficient enough to get control of the CPU, and **starvation can be avoided.**

➢Inter Process Communication

- Processes running into the system can be divided into two categories:

1. **Independent Processes:** - Process which do not shares data (i.e. resources) with any other process reffered as an independent process. OR Process which do not affects or not gets affected by any other process reffered as an independent process.
2. **Co-operative Processes: -** Process which shares data (i.e. resources) with any other process referred as co - operative process. OR Process which affects or gets affected by any other process referred as co-operative process.

❑Q. Why there is need of an IPC?

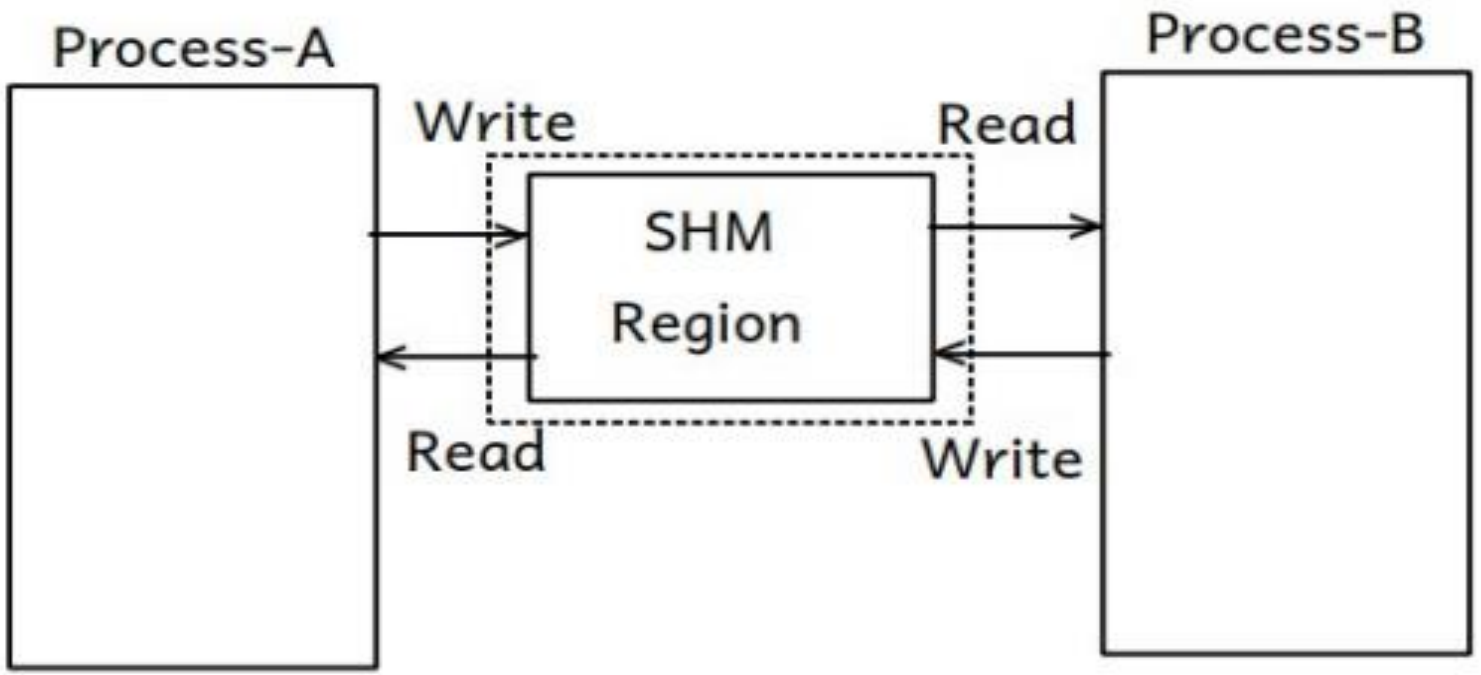
As concurrently executing co-operative processes shares common resources, so there are quite chances to occur conflictions between them and to avoid this conflictions there is a need of communication takes place between them.

➢What is an Inter Process Communication?

- An IPC is one of the important **service made available by the kernel, by using which co-operative processes can communicates with each other.**
- Inter process communication takes place only between co-operative processes.
- Any process cannot directly communicates with any other process, hence there is a need of some medium, and to provide this medium is the job of an OS/Kernel.
- There are two techniques by which IPC can be done/there are two IPC Models:

1. **Shared Memory Model:** under this technique, processes can communicates with each other by means of reading and writing data into the shared memory region (i.e. it is a region/portion of the main memory) which is provided by an OS temporarily on request of processes want to communicates.
2. **Message Passing Model:** under this technique, processes can communicates with each other by means of sending messages.

- Any process cannot directly send message to any other process.
- **Shared Memory Model is faster than Message Passing Model.**



SHARED MEMORY MODEL

➢Inter Process Communication

2. Message Passing Model:

there are further different IPC techniques under message passing model.

i. Pipe:

- By using Pipe **mechanism one process can send message to another process,** vice versa is not possible and **hence it is a unidirectional communication technique.**
- In this IPC mechanism, from one end i.e. from **write end** one process can writes data into the pipe, whereas from another end i.e**. from read end**, another process can read data from it, and communication takes place.
- There are two types of pipes:

1. **unnamed pipe:** in this type of pipe mechanism, only related processes can communicates by using pipe (|) command.
2. **named pipe:** in this type of pipe mechanism, related as well as non-related processes can communicates by using pipe() system call.

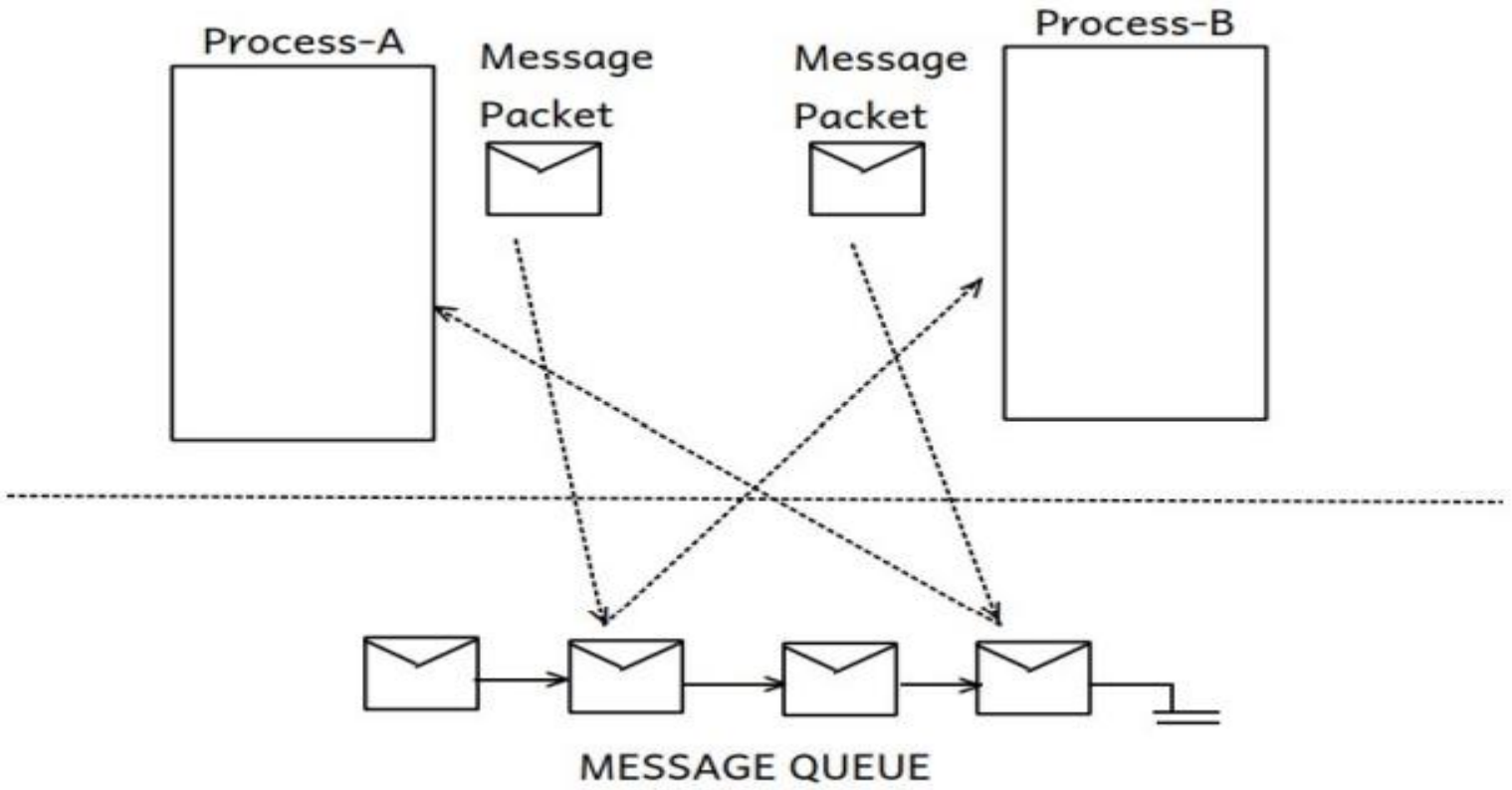
- By using Pipe only processes which are running in the same system can communicates,

➢Inter Process Communication

ii. Message Queue:

- By using message queue technique, processes can communicates by means of sending as well as receiving **message packets** to each other via **message queue** provided by the kernel as a medium, and hence it is a **bidirectional communication**.

- **Message Packet:** Message Header(Information about the message) + Actual Message.
- Internally an OS maintains message queue in which message packets sent by one process are submitted and can be sent to receiver process and vice-versa.
- By using message queue technique, only processes which are running in the same system can communicates.


Process-A
Message
Packet
Message
Packet
Process-B
MESSAGE QUEUE

➢Inter Process Communication

iii. Signals:

- Processes communicates by means of sending signals as well. - One process can send signal to another process through an OS.
- An OS sends signal to any process but any other process cannot sends signal to an OS.

Example: When we shutdown the system, an OS sends **SIGTERM** signal to all processes, due to which processes gets terminated normally, but few processes can handle SIGTERM i.e. even **after receiving this signal from an OS they** continues execution, to such processes an OS sends **SIGKILL** signal due to which processes **gets terminated forcefully.**

e.g. SIGSTOP, SIGCONT, SIGSEGV etc...

- - By using signal ipc technique, only processes which are running in the same system can communicates.

➢Inter Process Communication

iv. Socket

- Limitation of above all **IPC techniques is, only processes which are running on the same system can communicates, so to overcome this limitation Socket IPC mechanism has been designed.**
- By using socket IPC mechanism**, process which is running on one machine can communicates with process running on another machine, whereas both machines are at remote distance from each other and provided they connected in a network (either LAN / WAN/Internet).**
- **Socket = IP Address + Port Number.**

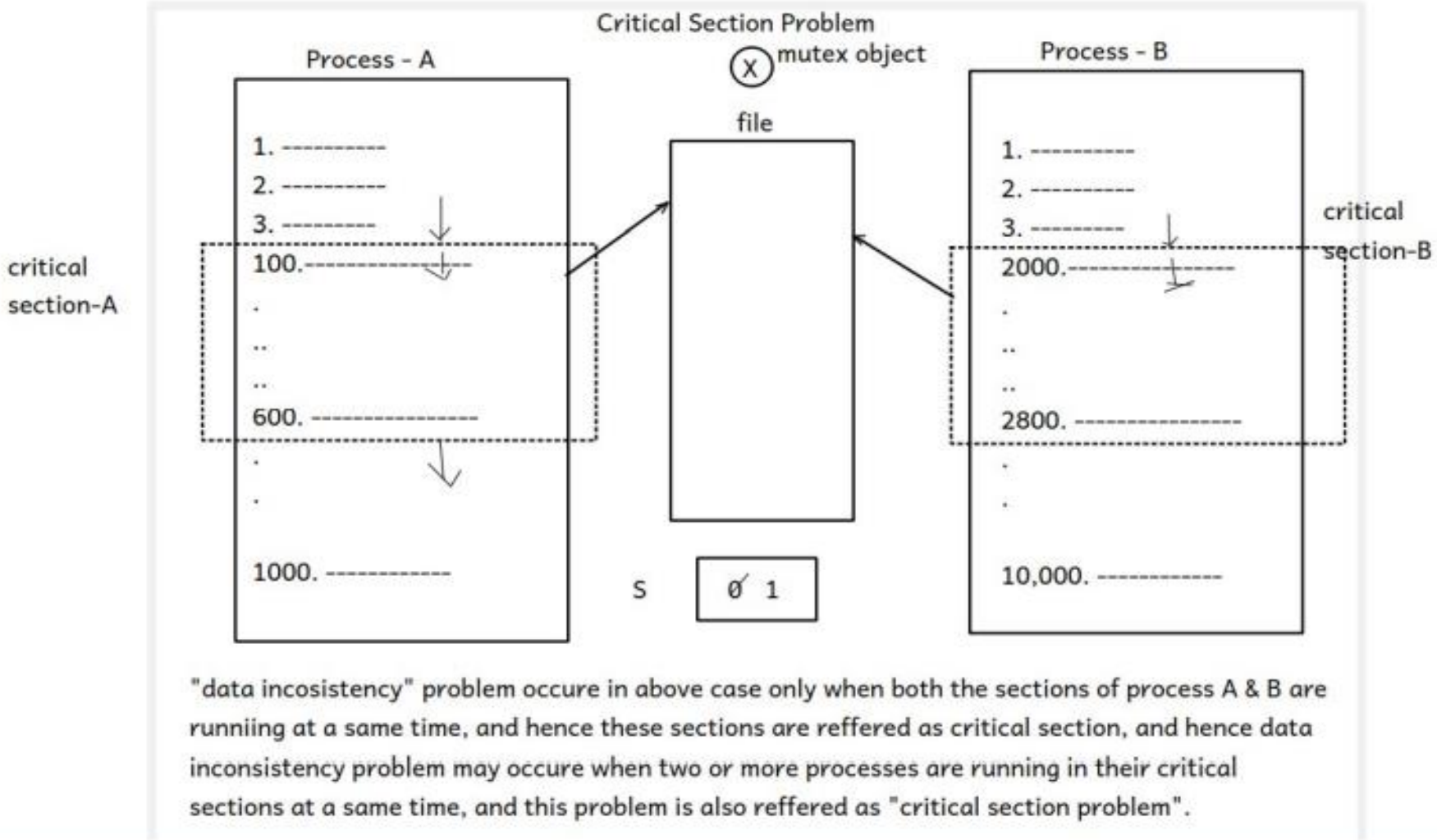
e.g. chatting application.

➢ Process Coordination / Process Synchronization

❑ Why Process Co-ordination/Synchronization?

- If concurrently executing co-operative processes are accessing common resources, then conflictions may takes place, which may results into the problem of **data inconsistency**, and hence to avoid this problem co-ordination / synchronization between these processes is required.
- **Race Condition:** if two or more processes are trying to access same resource at a time, race condition may occurs, and data inconsistency problem may takes place due to race condition.
- **Race condition** can be avoided by an OS by

1. deciding order of allocation of resource for processes, and
2. whichever changes did by the last accessed process onto the resource remains final changes.



"data incosistency" problem occure in above case only when both the sections of process A & B are runniing at a same time, and hence these sections are reffered as critical section, and hence data inconsistency problem may occure when two or more processes are running in their critical sections at a same time, and this problem is also reffered as "critical section problem".

➢Synchronization Tools:

1. Semaphore:

- there are two types of semaphore

i. Binary semaphore : can be used when at a time resource can be acquired by only one process.

- It is an integer variable having either value is 0 or 1.

ii. Counting / Classic semaphore : can be used when at a time resource can be acquired by more than one processes

2. Mutex Object: can be used when at a time resource can be acquired by only one process.

- Mutex object has two states **: locked & unlocked**, and at a time it can be only in a one state either locked or unlocked.
- Semaphore **uses signaling mechanism**, whereas mutex object uses **locking and unlocking mechanism.**

➢Deadlock: -

- There are four **necessary and sufficient conditions to occur deadlock / characteristics of deadlock:**

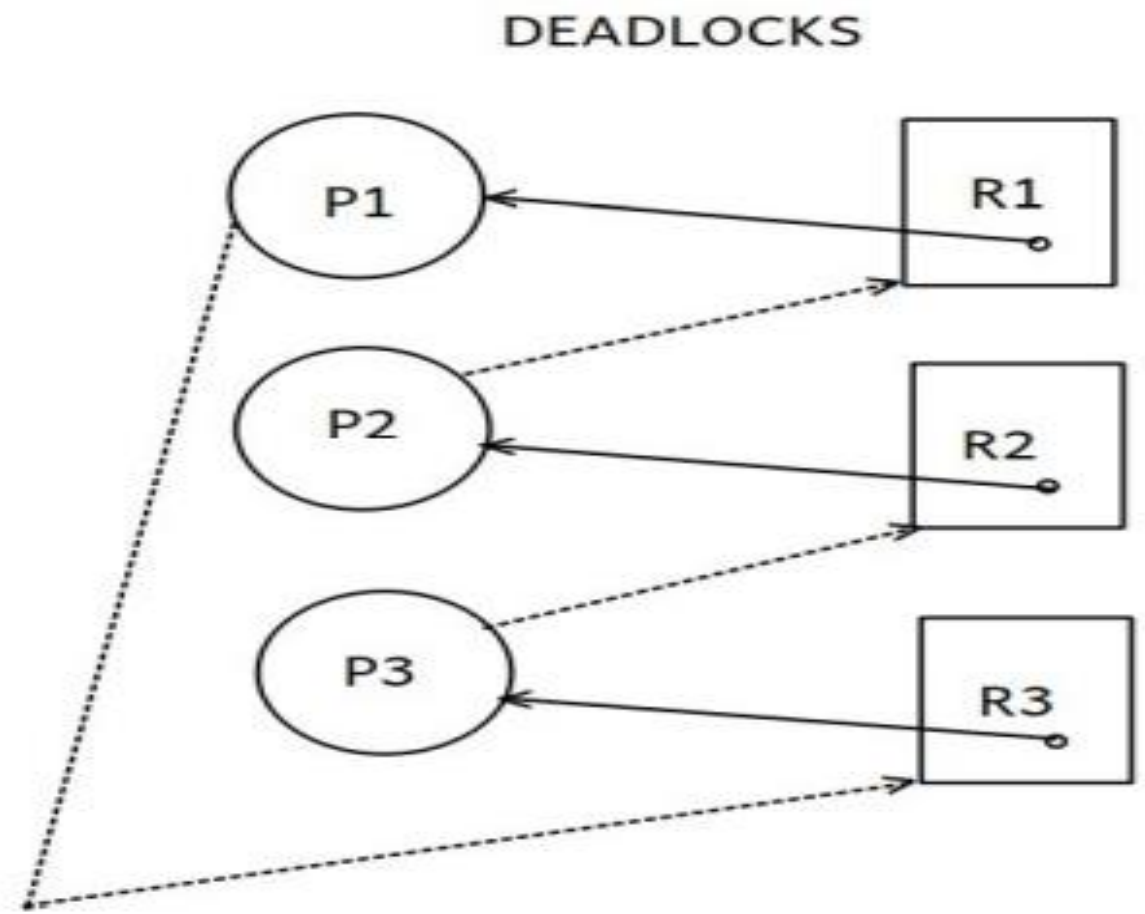
1. Mutual Exclusion: at a time resource can be acquired by only one process.

2. No Preemption: control of the resource cannot be taken away forcefully from any process.

3. Hold & Wait: every process is holding one resource and waiting for the resource which is held by another process.

4. Circular Wait: if process P1 is holding resource and waiting for the resource held by another process P2, and process P2 is also holding one resource and waiting for the resource held by process P1.

➢Deadlock: Resource Allocation Graph



➢Three deadlock handling methods are there:

1. Deadlock Prevention:

- deadlock can be prevented by discarding any one condition out of four necessary and sufficient conditions.

2. Deadlock Detection & Avoidance:

- before allocating resources for processes all input can be given to deadlock detection algorithm in advanced and if there are chances to occur deadlock then it can be avoided by doing necessary changes.

o There are two deadlock detection & avoidance algorithms:

1. **Resource Allocation Graph Algorithm**
2. **Banker's Algorithm**

3. Deadlock Recovery:

- System can be recovered from the deadlock by two ways:

1. Process termination: in this method randomly any one process out of processes causes deadlock gets selected and terminated forcefully to recover system from deadlock.

- Process which gets terminated forcefully in this method is referred as a victim process.

2. Resource preemption: in this method control of the resource taken away forcefully from a process to recover system from deadlock.

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com