# Operating Systems and Computer Fundamentals

## ➢Multi-level queue :

- In modern OS, the ready queue can be divided into multiple sub-queues and processes are arranged in them depending on their scheduling requirements. This structure is called as "Multi-level queue".
- If a process is starving in some sub-queue due to scheduling algorithm, it may be shifted into another sub-queue. This modification is referred as "Multi-level feedback queue".
- The division of processes into sub - queues may differ from OS to OS.

| | | |
|---|---|---|
| 1. | Important Services | :Priority Scheduling |
| 2. | Background Task | :SJF |
| 3. | GUI Tasks | :RR |
| 4. | Other Tasks | :FCFS |

## ➢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 referred as an independent process. OR Process which do not affects or not gets affected by any other process referred 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.

### ❖Reasons for cooperating processes:

- Information sharing
- Computation speedup
- Modularity
- Convenience
- Cooperating processes need inter process communication (IPC)

## 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.

## Q. 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.

# Operating Systems and Computer Fundamentals

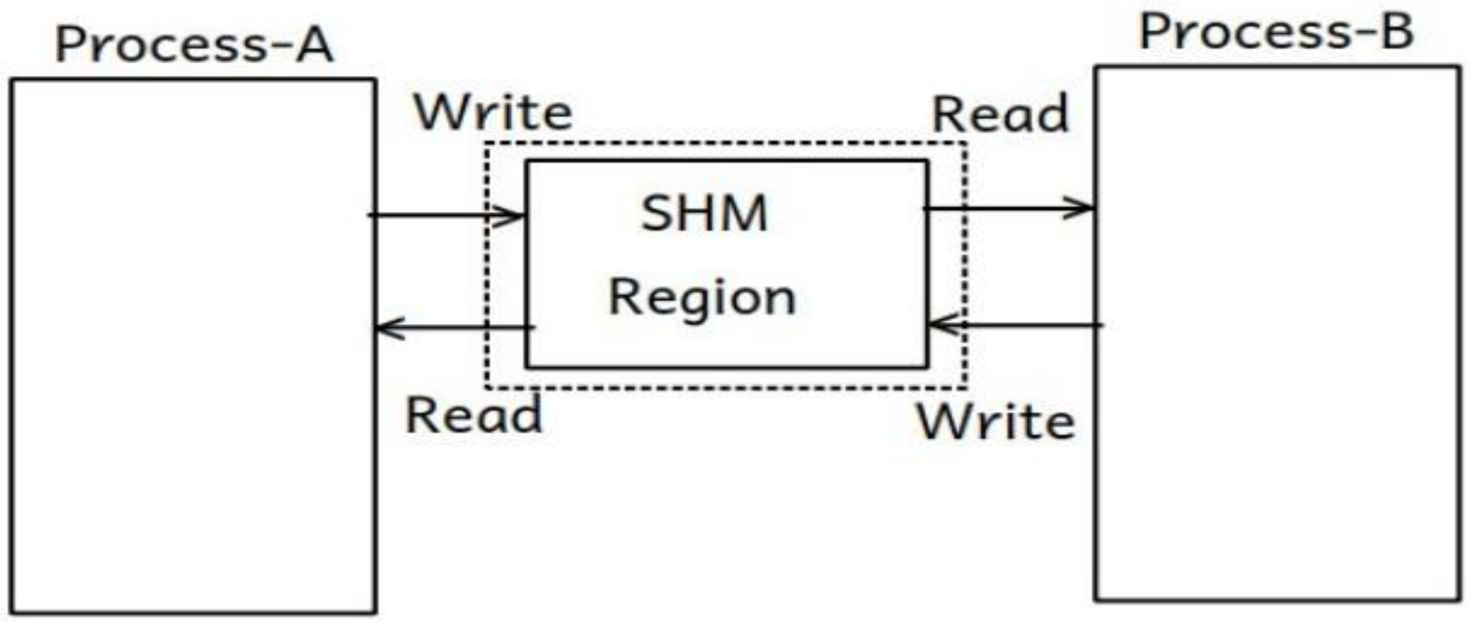
❖There are **two techniques** by which IPC can be done/there are two IPC Models:

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

## Operating Systems and Computer Fundamentals

-



SHARED MEMORY MODEL

**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:

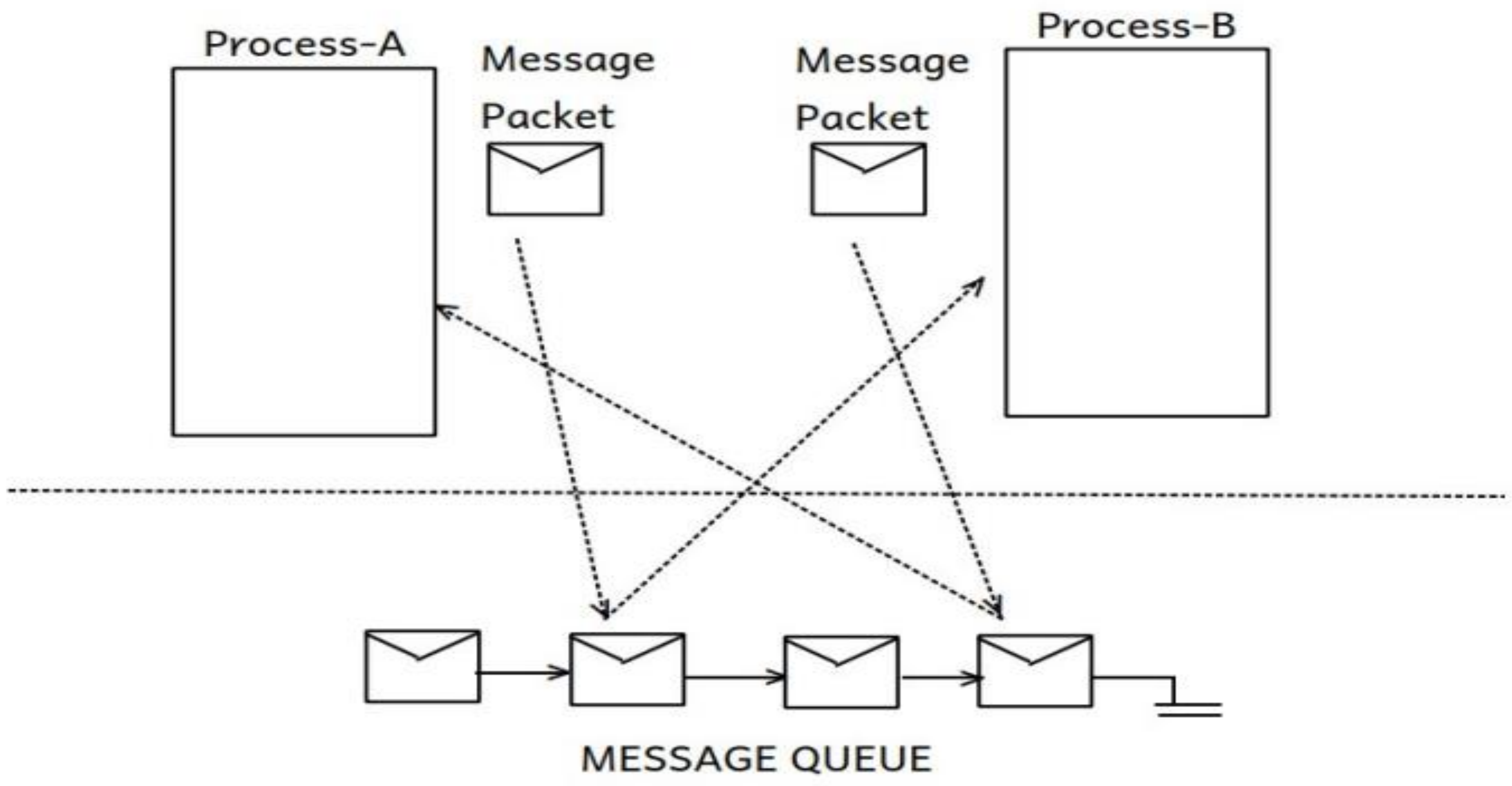
1. **unnamed pipe:** in this type of pipe mechanism, only related processes can communicates by **using pipe ( | ) command.**
2. **named pipe:** in this type of pipe mechanism, related as well as non-related processes can communicates by **using pipe() system call.**

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

## 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.

## Inter Process Communication



## iii. Signals:

- o Processes communicates by means of sending signals as well.
- o One process can send signal to another process through an OS.
- o An OS sends signal to any process but any another 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...**

# Operating Systems and Computer Fundamentals

## ❑Important Signals

- **SIGINT (2):** When CTRL+C is pressed, INT signal is sent to the foreground process.
- **SIGKILL (9)**: During system shutdown, OS send this signal to all processes to forcefully kill them. Process cannot handle this signal.
- **SIGSTOP (19)**: Pressing CTRL+S, generate this signal which suspend the foreground process. Process cannot handle this signal.
- **SIGCONT (18):** Pressing CTRL+Q, generate this signal which resume suspended the process
- **SIGSEGV (11):** If process access invalid memory address (dangling pointer), OS send this signal to process causing process to get terminated. It prints error message "Segmentation Fault".

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

## 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

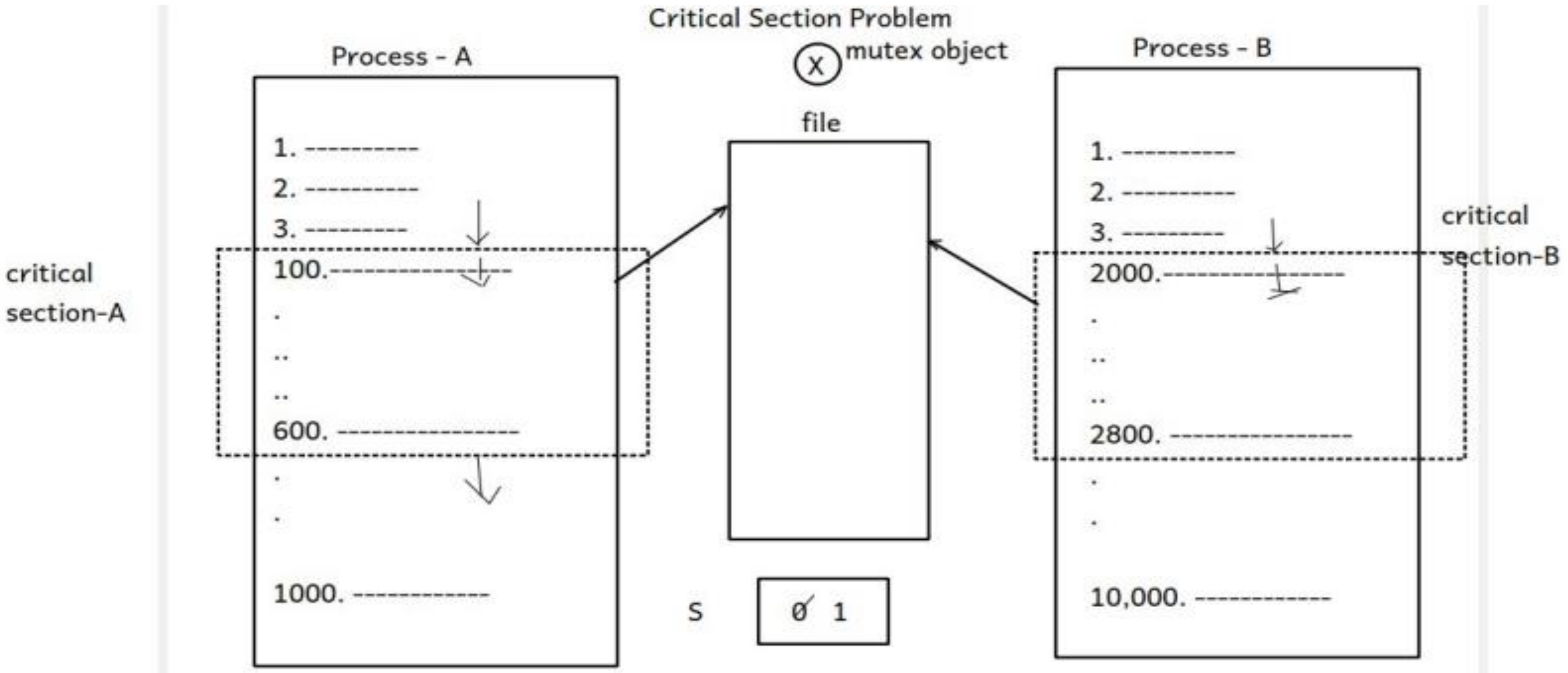
Why Process Co-ordination/Synchronization?

- 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 coordination / 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 .
2. whichever changes did by the last accessed process onto the resource remains final changes.

# Operating Systems and Computer Fundamentals



"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".

## Operating Systems and Computer Fundamentals

➢**Synchronization Tools**:

1. **Semaphore:**
2. **Mutex :**

➢**Semaphore:**

- Semaphore was suggested by Dijkstra scientist (dutch math)
- Semaphore is a counter

❑**On semaphore two operations are supported:**

➢**wait operation: decrement op: P operation:**

1. Semaphore count is decremented by 1.
2. If cnt < 0, then calling process is blocked(block the current process).
3. Typically wait operation is performed before accessing the resource.

# Operating Systems and Computer Fundamentals

## ➢signal operation: increment op: V operation:

1. semaphore count is incremented by 1.
2. if one or more processes are blocked on the semaphore, then wake up one of the process.
3. typically signal operation is performed after releasing the resource.

Q. If sema count = -n, how many processes are waiting on that semaphore?

Answer: "n" processes waiting

- 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

# Operating Systems and Computer Fundamentals

## 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

## ➢ Semaphore vs Mutex

S: Semaphore can be decremented by one process and incremented by same or another process.

M: The process locking the mutex is owner of it. Only owner can unlock that mutex.

S: Semaphore can be counting or binary.

M: Mutex is like binary semaphore. Only two states: locked and unlocked.

S: Semaphore can be used for counting, mutual exclusion or as a flag.

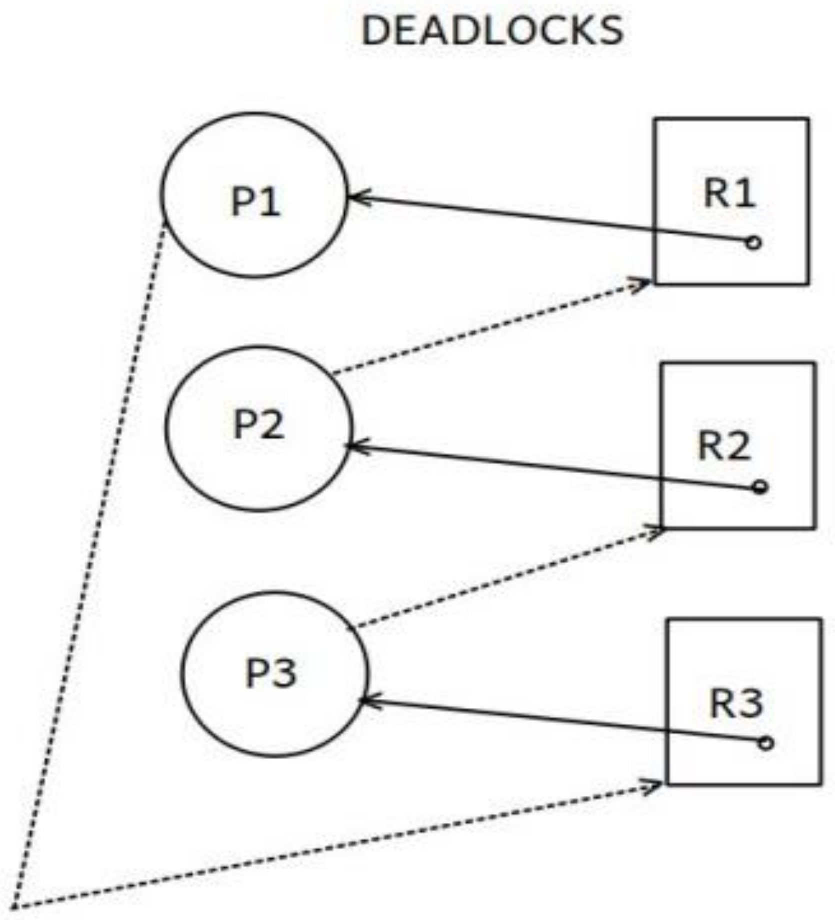
M: Mutex can be used only for mutual exclusion.

## ➢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.

## ❖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.

## ❑Starvation:

- The process not getting enough CPU time for its execution.
- Process is in ready state/queue. Reason: Lower priority (CPU is busy in executing high priority process).

## ❑Deadlock:

- The process not getting the resource for its execution.
- Process is in waiting state/queue indefinitely. Reason: Resource is blocked by another process (and there is circular wait).

- **System Calls:** are the functions defined in a C, C++ & Assembly languages, which provides interface of  services made available by the kernel for the user (programmer user).

- If programmers want to use kernel services in their programs, it can be called directly through system calls or indirectly through set of libary functions provided by that programming language.
- System calls internally use Software interrupt/Trap instruction to change CPU mode.
- System calls are different for different operating systems.
  - UNIX - 64 syscalls
  - Linux - 350+ syscalls
  - Windows - 3000+ syscalls

## -There are 6 categories of system calls:

## 1. Process Control System Calls: e.g. fork(), _exit(), wait() etc...

1. fork() : To create new processes
   - Creates a new process duplicating the calling process.
   - The new process is referred as child process, while the calling process is referred as parent process
2. exec() ~~_exit~~() : To load a new program (executable file) into current process's memory space.
3. wait() : To hold/wait processes

## 2. File Operations System Calls: e.g. open(), read(), write(), close() etc...

1. open() :
2. read() :
3. write() :
4. close() :

## 3. Device Control System Calls:

e.g. open(), read(), write(), ioctl() etc...

4. Accounting Information System Calls:

   e.g. getpid(), getppid(), stat() etc...

   1. getpid()   : To get Process ID
   2. getppid() : To get Parent Process ID
   3. stat()       : To get File Information

5. Protection & Security System Calls:

   e.g. chmod(), chown() etc..

   1. chmod() : Change user mode / permission
   2. chown() : get file owner info

6. Inter Process Communication System Calls:

   e.g. pipe(), signal(), msgget() etc...