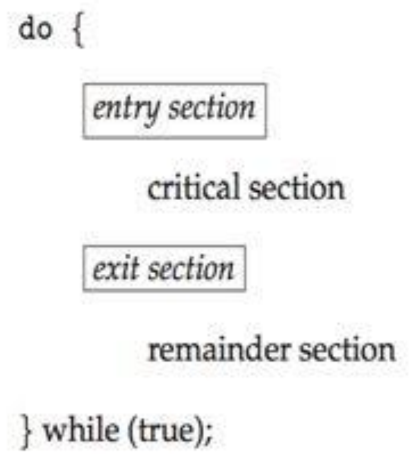
PROCESS SYNCHRONIZATION AND CPU SCHEDULING

CRITICAL SECTION

- That part of the program where the **shared memory is accessed** is called critical section. This part may not be concurrently executed by more than one process at a time. Each process must ask permission to enter critical section with ENTRY section and then with EXIT section, comes out of that and works with remainder section. Following code snippet shows the access of shared memory by process:



```
do {
    entry section
        critical section
    exit section
        remainder section
} while (true);
```

There are 3 principles involved in dealing with critical section:

1. MUTUAL EXCLUSION

- If one process is executing in critical section, no other process is allowed entry to that.

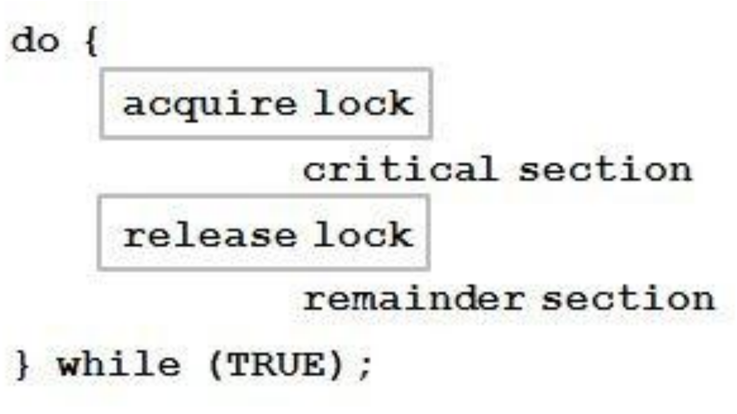
2. PROGRESS

- If no process is currently in critical section and some processes want to enter then only those processes which are not executing in their remainder section can participate in decision making that which one will enter the critical section next.

3. BOUNDED WAITING

- Once a process enters critical section, it cannot enter again until a waiting process gets its turn. Entry is managed as a queue.

- Many systems even provide hardware support for implementing critical section code. They are based on idea of locks. **The algorithm for using locks looks something like as below:**



```
do {
    acquire lock
            critical section
    release lock
            remainder section
} while (TRUE);
```

SEMAPHORES

- A semaphore is a protected integer variable that can facilitate and restrict access to shared resources in multi programming environment. They were invented by **Edsger Djikstra**. Two most common kinds of semaphores are Binary and Counting semaphores.
- **Counting semaphores** represent multiple resources and **binary semaphore** represent two possible states i.e 0 or 1 (locked and unlocked). They are accessed by two operations i.e **Wait ( ) and Signal( ).**

DEADLOCK AND STARVATION

- **Deadlock** is a situation where two or more processes are **waiting indefinitely** for an event that an be caused by only one of the waiting process.
- **Starvation** is a situation where a process may **never be removed** from semaphore queue in which it is suspended.

CPU SCHEDULING

- CPU scheduler selects from among the processes in ready queue which are to be allocated CPU next. CPU scheduling decision takes places when a process moves through following states:
    - Running to Waiting
    - Running to Ready
    - Waiting to Ready
    - Termination
- Scheduling for all of above transitions is non-preemptive in nature. All of rest is preemptive in nature. **Non Preemptive scheduling** is when CPU is given to a process; it cannot be taken away unless process finishes execution where as **preemptive scheduling** is when a high priority task is allocated the CPU while interrupting the currently running process. It is based on idea that highest priority process should always be the process that is currently utilized.

SCHEDULING CRITERIA FOR PROCESSES

As we know that CPU scheduler selects from among the processes in ready queue and assigns CPU to one of them. This scheduling is done on basis on certain parameters **called scheduling criteria** which are as follows:

- **CPU utilization -** aim is to keep CPU as busy as possible.
- **Throughput -** It is viewed in terms of number of processes that complete their execution per time unit.
- **Turnaround Time -** amount of time a process takes to execute.
- **Waiting Time -** amount of time a process has been waiting in the ready queue.
- **Response Time -** amount of time it takes for first response after a request has been submitted.

SCHEDULING ALGORITHMS

The algorithms used mostly in practice are discussed as follows:

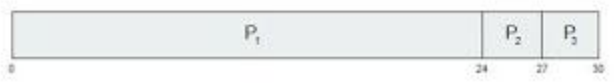
1. FIRST COME FIRST SERVED SCHEDULING (FCFS)

Major attributes of FCFS are as follows :

- It queues processes in the order that they arrive in ready queue.
- Performance is usually poor as **average response time is high.**

Consider the process below & their burst times and lets see how their scheduling charts looks like. **(Burst time -** amount of CPU time a process requires).

Process	Burst Time
P_1	24
P_2	3
P_3	3

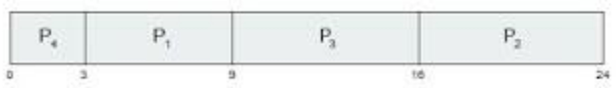


2. SHORTEST JOB FIRST SCHEDULING (SJF)

Major attributes of SJF are as follows:

- Scheduler picks processes according to their burst times. Shortest one is picked first followed by next shortest one.
- It gives minimum average waiting time for a given set of processes.

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



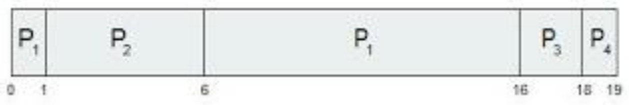
3. PRIORITY SCHEDULING

Major attributes are as follows:

- A priority number is associated with each process (an integer value)
- CPU is allocated to the process with highest priority.

Problem - low priority process may never execute. Solution to that would be to increase the priority as time progresses for a ready process.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

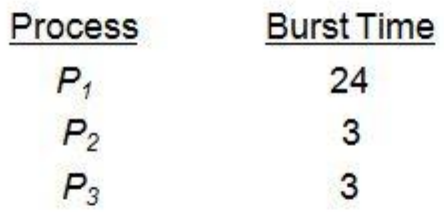


4. ROUND ROBIN SCHEDULING

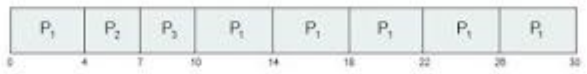
Major attributes are as follows:

- Each process gets a small unit of CPU time called quantum.
- After that time, current process is preempted and added to end of ready queue. Next process gets CPU for next quantum.
- It has **higher avg waiting time** than SJF but better response.

Example of Round Robin Scheduling with time quantum = 4 is as below:



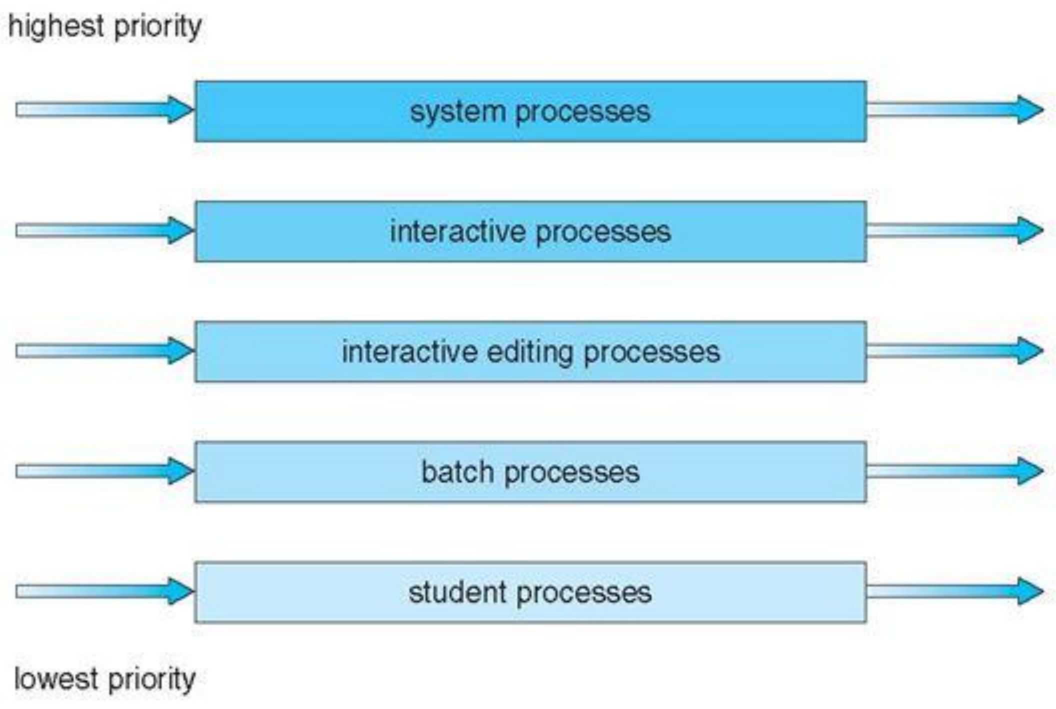
Process	Burst Time
P_1	24
P_2	3
P_3	3



5. MULTI LEVEL QUEUE SCHEDULING

Major attributes are as follows:

- In this, ready queue is partitioned into separate queues e.g queue 1 and queue 2.
- One process remains permanently in one queue and can not shift queues.
- Each queue can have its own scheduling algorithms.
- Scheduling must be done between two queues and once CPU assigned to a queue, process in that will be executed before moving to another queue.



6. THREAD SCHEDULING

Major attributes are as follows:

- Scheduler assigns CPU based on a distinction between user level threads and kernel threads.
- Competition is between threads inside a process for allocation of CPU.

DISPATCHER

- Dispatcher gives **control of CPU** to process selected by the scheduler. It involves **switching context**, switching to user mode and jumping to proper location to restart the program. **Dispatch latency** is defined as the time it takes for dispatcher to stop one process and start another.