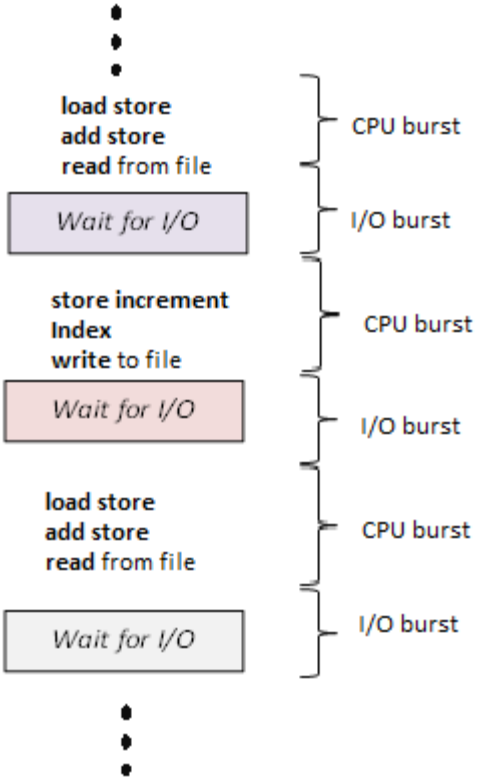
CPU SCHEDULING

CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.
In a single – processor system, only one process can run at a time, any others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
The process of CPU scheduling depends on an observed property of processes: process execution consists of a cycle of CPU execution and I/O wait. Processes alternate between these two states. Process execution begins with a **CPU burst**. That is followed by an **I/O burst**, which is followed by another CPU burst, then another I/O burst, and so on.



Types of scheduling

Preemptive Scheduling: Preemptive scheduling includes the following four execution states:-
1. When process switches from the running state to the waiting state.
2. When a process switches from the running state to the ready state.
3. When a process switches from the waiting state to the ready state.
4. When a process terminates.

Non – Preemptive Scheduling: When scheduling take place only under circumstances 1 and 4 i.e. When process switches from the running state to the waiting state and when a process terminates, we say that the scheduling scheme is non – preemptive.

Scheduling Criteria:

The criteria include the following:

- **CPU utilization:** We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent to 90 percent.
- **Throughput:** The number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be ten processes per second.
- **Turnaround Time:** The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
- **Waiting time:** Waiting time is the sum of the periods spent waiting in the ready queue.
- **Response time:** The time from the submission of a request until the first response is produced, is termed as response time. Response time, is the time it takes to start responding, not the time it takes to output the response.

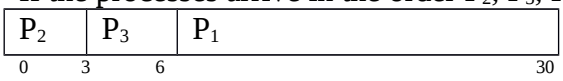
Scheduling Algorithm

1. **First – Come , First – Served (FCFS) Scheduling:** The simplest CPU – scheduling algorithm is the FCFS scheduling. With this scheme ,the process that request the CPU first is allocated the CPU first. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.
   On the negative side ,the average waiting time under the FCFS policy is often quite long. Consider the following set of process that arrive at time 0, with the length of the CPU burst given in milliseconds;

Process	Burst Time
P_1	24
P_2	3
P_3	3

   If the processes arrive in the order P_1, P_2, P_3 , and are served in FCFS order, we get the result Gantt chart

   The waiting time is 0 milliseconds for process P_1, 24 milliseconds for process P_2, and 27 milliseconds for process P_3.
   Average waiting time = (0+24+27)/3=17
   If the processes arrive in the order P_2, P_3, P_1, then



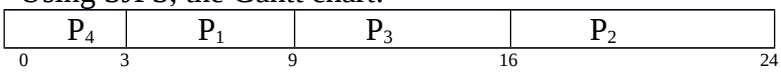
   The average waiting time = (6+0+3)/3=3 milliseconds.
   The FCFS scheduling is non – preemptive. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU ,either by terminating or by requesting I/O. The FCFS algorithm is thus particularly troublesome for time – sharing systems, where it is important that each user get a share of the CPU at regular intervals.
2. **Shortest-Job-First Scheduling (SJFS) :** This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the

process that has the smallest next CPU burst. If the next CPU burst of two processes are the same, FCFS scheduling is used to break the tie.

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Using SJFS, the Gantt chart:



The waiting time is 3 milliseconds for process P_1, 16 milliseconds for process P_2, 9 milliseconds for process P_3, and 0 milliseconds for process P_4.

The average waiting time = (3+16+9+0)/4 = 7

The SJFS algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes. Moving a short process before a long one decreases the waiting time of the short process more than it increases the waiting time of the long process. Consequently, the average waiting time decreases.

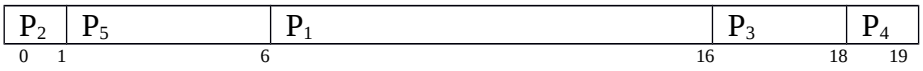
The real difficulty with the SJF algorithm is knowing the length of the next CPU request.

3. **Priority Scheduling:** A priority scheduling is associated with each process, and the CPU is allocated to the process with the highest priority. Equal – priority processes are scheduled in FCFS order. An SJF algorithm is simply a priority algorithm where the priority(*P*) is the inverse of the next CPU burst. The larger the CPU burst, the lower the priority, and vice versa.

   Consider the following set of processes, assumed to have arrived at time 0 in the order P_1, P_2, …, P_5, with the length of the CPU burst given in milliseconds:

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

   Gantt chart:



   Average waiting time = 8.2.

Priority Scheduling can be either preemptive or non – preemptive. When a process arrives at the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. A non-preemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.

A major problem with the priority scheduling algorithms is indefinite blocking or starvation. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low-priority processes waiting indefinitely.

4. **Round – Robin Scheduling:** The round-robin (RR) scheduling algorithm is designed especially for time – sharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue. the CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
   To implement RRS, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. the CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.
   The average waiting time under the RR policy is often long.
   Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds.

Process	Burst Time
P_1	24
P_2	3
P_3	3

   If we use a time quantum of 4 milliseconds, then process P_1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is preempted after first time quantum, and the CPU is given to the next process in the queue, process P_2. Process P_2 does not need 4 milliseconds, so it quite before its time quantum expires. The CPU is then given to the next process, process P_3. Once each process has received 1 time quantum, the CPU is returned to process P_1, for an additional time quantum. The resulting RR schedule is as follows:

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1
0 4	7	10	14	18	22	26	30

   P_1 waits for 6 milliseconds(10-4), P_2 waits for 4 milliseconds, and P_3 waits for 7 milliseconds.
   Average waiting time = 5.66 milliseconds.
5. **Multilevel Queue Scheduling:** A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned tp one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm.

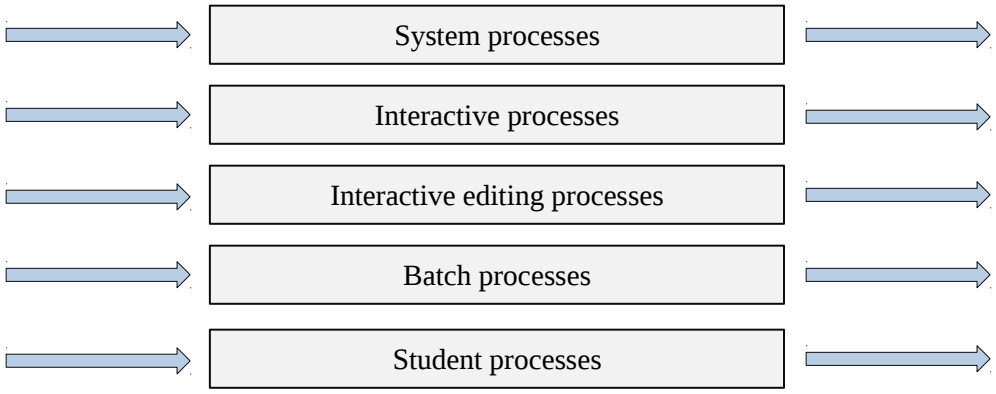


Fig: Multilevel queue scheduling