SCHEDULING :- Scheduling is the activity of determining which service request should be handled by a system. Scheduling is important because it influences user services and efficiency of CPU utilization.

Scheduling refers to a set of policies and mechanisms supported by operating system that controls the order in which the work to be done is completed and selects the next job to be admitted for execution.

Types of Schedulers :- There are three types of schedulers :-

(i) Long Term schedulers / Job scheduler
(ii) Medium term scheduler
(iii) Short term schedulers / CPU scheduler

Long Term Scheduler :- Sometimes it is also called job scheduling. This determines which job shall be submitted for immediate processing i.e. jobs submitted to the system become part of an input queue; a job scheduler selects jobs from this workload.

There are always more processes that it can be executed by CPU as in Batch Operating System. These processes are kept in large

storage devices like disk for later processing. The long term scheduler selects processes from this pool and loads them into memory. In memory these processes belong to a ready queue. Queue is a type of data structure.

Short term scheduler :- It allocates processes belonging to ready queue to CPU for immediate processing. Its main objective is to maximize CPU requirement. In CPU scheduling, this information may include the following:-

1) Size of a request in a CPU seconds.
2) CPU time already consumed by a request.
3) Deadline by which its results are required.

The short term scheduler (also called the CPU scheduler), selects from among the processes in memory which are ready to execute and assigns the CPU to one of them. Most processes can be put into any of two categories:-

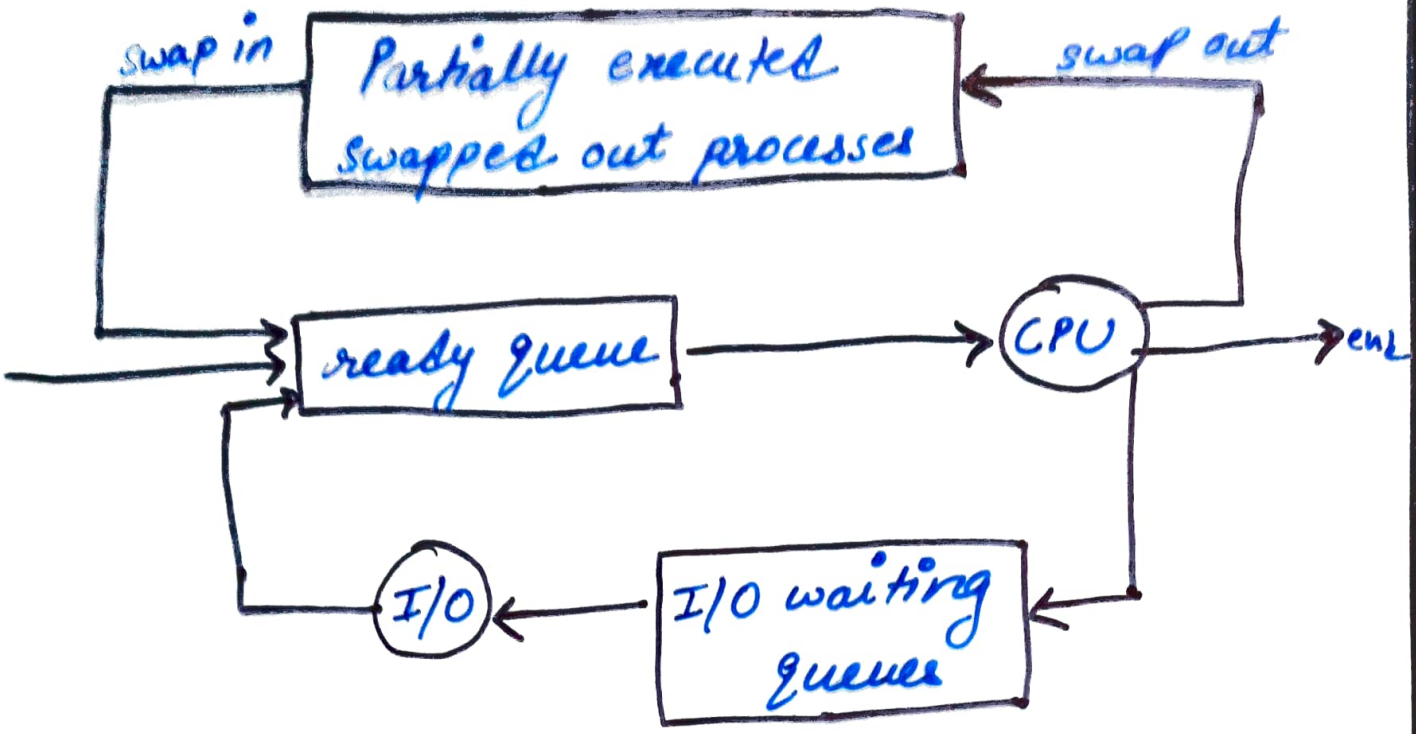
→ CPU Bound
→ I/O Bound

If all processes are I/O bound, the ready queue will always be empty & the short term scheduler will have nothing to do. If all processes are CPU bound the process will be waiting for I/O operation and again the system will be unbalanced. Therefore

the long term scheduler provides good performance by selecting combination of CPU bound and I/O bound process.

Medium term scheduler :- Most of the processes require some I/O operation. In that case, it may become suspended for I/O operation after running a while. It is beneficial to remove these process (suspended) from main memory to hard disk to make room for other processes. At some later time these processes can be reloaded into memory and continued where from it was left earlier. saving of the suspended process is said to be swapped out or rolled out. The process is swapped in and swap out by the medium term scheduler.

The medium term scheduler has nothing to do with the suspended processes. But the moment the suspending condition is fulfilled, the medium term scheduler get activated to allocate the memory and swap in the process and make it ready for commenting CPU resource.



Note :- The primary distinction between long term & short term scheduler lies in frequency of execution. The short term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short term scheduler executes atleast once every 100 milliseconds. Because of the short time between executions, the short term scheduler must be fast. If it takes 10 milliseconds to decide to execute a process for 100 milliseconds, then $10/(100+10) = 9$ percent of the CPU is being used simply for scheduling the work.

The long term scheduler controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average

rate of process creation must be equal to the average departure rate of processes leaving the system. Thus, the long term scheduler may need to be invoked only when a process leaves the system. Because of the longer interval between executions, the long term scheduler can afford to take more time to decide which process should be selected for execution.

CPU Scheduling :- Several processes are kept in memory at one time. When one process has to wait, the OS takes the CPU away from that process and gives the CPU to another process.

Scheduling of this kind is a fundamental OS function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources.

Note :- The ready queue is not necessarily a first-in, first-out (FIFO) queue. A ready queue may be implemented as a FIFO queue, a priority queue, a tree or simply queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally PCBs of the processes.

Preemptive Scheduling :- CPU scheduling decisions may take place under the following four circumstances:

1) when a 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.

When scheduling takes place only under circumstances 1 and 4, we say that the scheduling scheme is non preemptive or cooperative; otherwise it is preemptive. Under non preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. eg - windows 3.x, windows 95 & soon introduces preemptive scheduling.

Dispatcher :- Another component involved in the CPU scheduling function is the dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short term sch.

This function involves the following:

- ⇒ switching context
- ⇒ switching to user mode
- ⇒ Jumping to the proper location in the user program to restart that program.

The time it takes for the dispatcher to stop one process and start another running is known as the dispatcher latency.

Scheduling Criteria :- Different CPU scheduling algorithms have different properties and may favor one class of processes over another. Many criteria have been suggested for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

(i) CPU Utilization: We want to keep the CPU as busy as possible.

(ii) Throughput: If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput.

(iii) Turnaround time: From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the

time of submission of a process to the time of completion is the turnaround time.

Waiting time: The CPU scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

Response time: is the time from the submission of a request until the first response is produced. This measure called response time, is the time it takes to start responding, not the time it takes to output the response.

Scheduling Algorithms

(i) FCFS

Process	Burst Time
P1	24
P2	3
P3	3

(ii) SJF

①

Process	Burst time
P1	6
P2	8
P3	7
P4	3

②

Process	Arrival	Burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

(v) Priority Scheduling :-

	Time	Priority
P1	5	3
P2	4	2
P3	6	1
P4	2	4

③

	ET	Pri:	AT
P_0	5	3	0
P1	4	1	1
P2	6	2	2
P3	2	1	2

	BT	AT	Priority
P1	1	0	1
P2	1	0	2
P3	3	2	3
P4	2	2	4
P5	1	3	4
P6	1	4	5

ROUND ROBIN SCHEDULING :-

Process	Burst time
P_1	24
P_2	3
P_3	3

Time quantum = 4 milli sec.

Assignment

	BT	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5 all at time 0.

a) Draw Gantt charts using FCFS, SJF, RR(q=1), non preemptive priority (SN implies a higher priority)

b) Find out Turn around time, waiting time

c) Minimal average waiting time (over all process)?

A major problem with priority scheduling algo 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. In a heavily loaded computer system, a steady stream of higher priority processes can prevent a low priority process for ever getting the CPU.

A solution to the problem of indefinite blockage of low-priority processes is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time. For example, if priorities range from 127 (low) to 0 (high), we could increment the priority of a waiting process by 1 every 15 minutes.

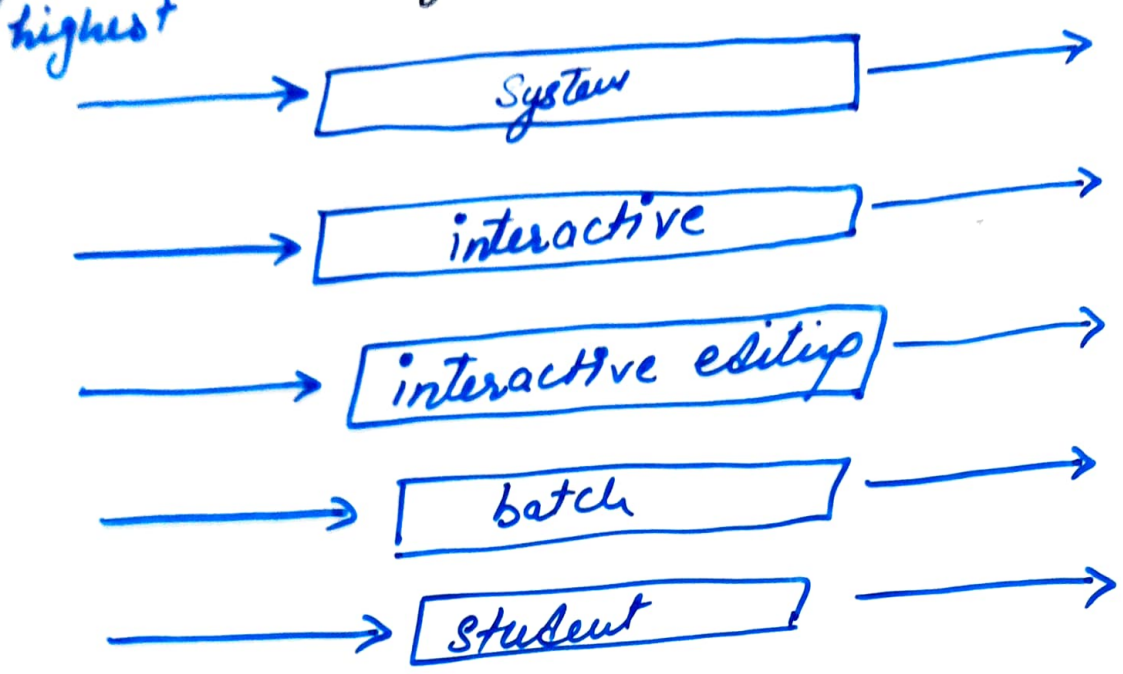
Q what is the effect of size of time quantum on the performance of R.R. CPU Scheduling Algo.

The performance of the RR algorithm depends heavily on the size of the time quantum. At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy. If the time quantum is extremely small (say 1 millisecond), the RR approach is called processor sharing & creates the appearance that each of n processes has its own processor running at 1/n the speed of the real processor.

Multilevel Queue Scheduling :- Processes are easily classified into different groups. A common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements and so might have different scheduling needs.

A multilevel queue-scheduling algo. partitions the ready queue into several separate queues. The processes are permanently assigned to 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. For example, separate queues might be used for foreground and background processes. The foreground might be scheduled by an RR alg; while the background queue is scheduled by an FCFS algorithm.

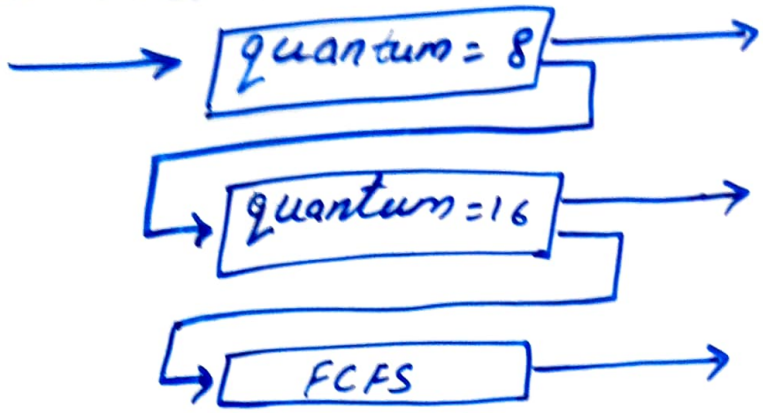


Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process would be preempted.

Another possibility is to time slice among the queues. Each queue gets a certain portion of the CPU time, which it can then schedule among the various processes in its queue.

Multilevel Feedback Queue Scheduling :- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues.

Multilevel feedback queue scheduling, in contrast, allows a process to move between queues. If a process uses too much CPU time, it will be moved to a lower priority queue. This scheme leaves I/O bound and interactive processes in the higher priority queues. In addition, a process that waits too long in a lower priority queue may be moved to a higher priority queue. This form of aging prevents starvation.



A process entering the ready queue is put in queue 0. A process in queue 0 is given a time quantum of 8 milliseconds. If it does not finish within this time, it is moved to the tail of queue 1. If queue 0 is empty, the process at the head of queue 1 is given a quantum of 16 milliseconds. If it

does not complete, it is preempted and is put into queue 2. Processes in queue 2 are run on an FCFS basis but are run only when queues 0 and 1 are empty.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

- ⇒ The number of queues.
- ⇒ The scheduling algorithm for each queue
- ⇒ The method used to determine when to upgrade a process to a higher priority queue.
- ⇒ The method used to determine when to demote a process to a lower priority queue.
- ⇒ The method used to determine which queue a process will enter when that process needs service.

Multiple Processor Scheduling:- Concentrate on systems in which the processors are identical — homogeneous — in terms of their functionality; we can then use any available processor to run any processes in the queue. limitations are - consider a system with an I/O device attached to a private bus of one processor. Process that wish to use that device must be scheduled to run on that processor.

If several identical processor are

available, then load sharing can occur. We could provide a separate queue for each processor. In such a scheme, one of two scheduling approaches may be used. One approach has all scheduling decisions, I/O processing and other system activities handled by a single processor - the master server. The other processors execute only user code. This asymmetric multiprocessing is simple because only one processor accesses the system data structures, alleviating the need for data sharing.

A second approach uses symmetric multiprocessing (SMP), where each processor is self scheduling. Each processor examines the common ready queue and selects a process to execute.

Real Time Scheduling :- Real time computing is of two types. Hard real time systems are required to complete a critical task within a guaranteed amount of time. Such a guarantee, made under resource reservation, requires that the scheduler know exactly how long it takes to perform each type of operating system function; therefore, each operation must be guaranteed to take a maximum amount of time.

Soft real time computing is less restrictive. It requires that critical processes receive priority over less fortunate ones.