CHAPTER 4

Scheduling

Syllabus

Scheduling – Objectives, concept, criteria, CPU and I/O burst cycle. Types of Scheduling-Pre-emptive, Non pre- emptive. Scheduling Algorithms. First come first served (FCFS), Shortest job first (SJF), Round Robin (RR), Priority. Other Scheduling. Multilevel, Multiprocessor, real-time. Deadlock. System model, principle necessary conditions, mutual exclusion, critical region. Deadlock handling. Prevention, avoidance algorithm-Banker's algorithm, Safety algorithm.

Statistical Analysis

BTE Paper	Marks allotted for this chapter
Winter 2008	30 Marks
Summer 2009	26 Marks
Winter 2009	28 Marks
Summer 2010	22 Marks
Winter 2010	18 Marks

- Process execution begins with CPU burst followed b I/O burst again CPU burst, I/O burst and so on. Th last state is executed by CPU with a system request i termination of execution.
- The duration of CPU burst is calculated extensivel The CPU burst time changes from process to process o even machine to machine. The frequency curve of CPU is shown in Fig. 4.2.
- The CPU burst curve is featured as exponential o hyper exponential with large number short time CPU bursts and small number of long time CPU executions.
- In case of I/O bound program, CPU bursts are small. I case of CPU bound programs, I/O bursts are long. This helps in designing scheduling algorithms.

4.1.4 CPU Scheduler

- We have already discussed that CPU should be busy al the time for the purpose of utilization. When the CPU is idle, an operating system should select the process from the ready queue and assign CPU to it.
- When ready queue is available, to select process from that queue for execution requires some selection criteria. There are different methods which decide the selection of process for execution. The short term scheduler or CPU schedulers selects an algorithm for process selection and picks up process from queue.
- It is not necessary that if queue of ready process is available then they will be assigned CPU in first come first serve basis, there could be different algorithms used by CPU schedulers. The selection algorithms could be FIFO, Priority, tree or unordered link list.

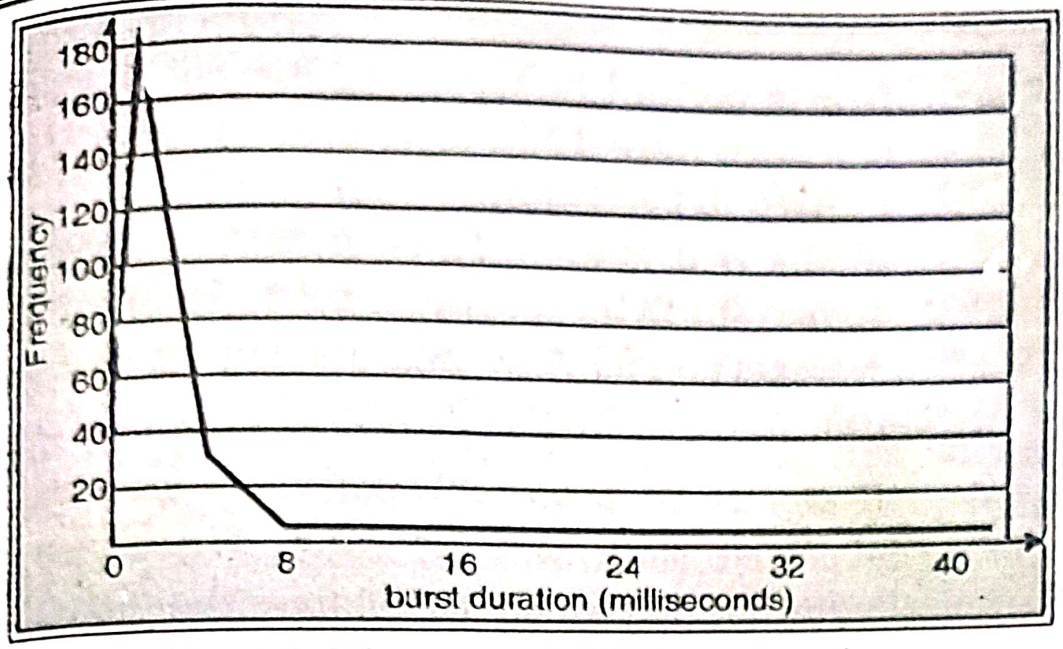

Fig. 4.2 : Histogram of CPU-burst duration

- CPU scheduling decisions may take place when a process :
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates

4.1.5 CPU Scheduling Criteria

Q. State and explain criteria in CPU scheduling. **W-08**

Q. Explain different process scheduling criteria. **S-09**

- As discussed earlier, there are different algorithms for CPU scheduling. Choosing an appropriate algorithm is very important. There are certain properties of algorithms which should be taken into consideration while algorithm selection.
- These properties are very useful to decide which schedule is best in which situation for the CPU. The criteria includes the following parameters

1. CPU utilization	2. Throughput
3. Turnaround time	4. Waiting time
5. Response time	

1. **CPU utilization**

- It is important to keep CPU always busy, so that it will be fully utilized. Theoretically CPU can be utilized in percentage from 0 to 100.
- In the real system, CPU utilization ranges from 40 percent to 90 percent. 40 % is for lightly loaded systems and 90 % for heavy loaded systems.

2. **Throughput**

- If we try to keep CPU busy all the time, then expected work will be done. There has to be certain unit which calculates the actual work done by CPU.
- There is one measure to calculate work done by CPU i.e. number of processes that are completed per time unit. This is called as **throughput**. If the processes are long then time could be one process per hour or for short processes it could be 10 processes per second.

3. **Turnaround time**

- Turnaround time is the time spent between the process submission and process completion. In short it is a time required for a process to get completely execute. So time between submission of process and completion of process is called as turnaround time.
- Turnaround time is sum of periods or time spent for waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing input/output operations

4. **Waiting time**

Generally the CPU scheduling algorithms do not get affected by the execution time of the processes or doing input/output operations

amount of time they spend waiting in the ready queue. The sum of the periods spent waiting in the ready queue is referred as waiting time.

Response time

- Turnaround time cannot be good criteria for selecting algorithm in an interactive system. Usually a process could produce some output quite early and can continue computing new results while previous results are being output to the user. So there is another measure to calculate the time between the actual submission of the process and till the first response is produced. This time is called as response time.
- This is a time which takes to start responding and the time it takes to output the response. The turnaround time is restricted by the speed of the output devices from the system.
- Ideally we can say that CPU utilization and throughput should be maximum and the remaining criteria like turnaround time, response time and waiting time should be minimized. But practically it may not be possible. So average values of these times could be a solution.
- Even in some cases minimum and maximum values can be used.

4.2 Types of Scheduling (Preemptive and Non-preemptive)

Q. What is pre-emptive and non pre-emptive scheduling? **S-09**

Q. Explain the pre-emptive and non-pre-emptive type of scheduling. State when pre-emptive and non pre-emptive scheduling should be used. **W-09**

Q. Describe the terms : **W-10**

(i) Pre-emptive scheduling

(ii) Non-pre-emptive scheduling

- The scheduling could be broadly classified into two types one is preemptive and other is non-preemptive. There are certain conditions on which scheduling are based and there we can decide which is preemptive and which is non-preemptive. The conditions are as follows:
  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.
- After reading conditions 1st and 4th we understand that there is no choice of scheduling. The process selection for execution comes in conditions 2nd and 3rd.
- The scheduling in case of 1st and 4th condition is called as non-preemptive or cooperative scheduling. The scheduling in 2nd and 3rd conditions is known as preemptive. In non-preemptive scheduling, when process is assigned a CPU, it keeps it busy till it gets terminate or till it switches to the waiting state. This method of non-preemptive scheduling was used by Windows 3.x version.
- In preemptive scheduling multiple processes can run. One process can be preempted to run another process.
- After this Windows 95 introduced preemptive scheduling. MAC operating system in its recent version has used non-preemptive scheduling or cooperative scheduling. But now a days the new versions have used preemptive scheduling.
- Non-preemptive scheduling can be used on certain specific platforms, as it does not require specific hardware. But unfortunately preemptive scheduling requires specific hardware and even it incurs high cost associated with access to shared data.
- Preemptive schedule influence the design of operating system kernel. Just imagine the situation where

kennel is busy with some system related task which is assigned by the system call. These activities may involved changing kernels important data. If due to preemptive schedule if kernel has to preempt this process just imagine what is going to happen. There will be a big chaos.

- This kernel model is very poor performing model to real time computing and multiprocessing operating systems.

Difference between pre-emptive and non-preemptive :

Sr. No.	Pre-emptive scheduling	Non-pre-emptive scheduling
1.	In pre-emptive scheduling a process switches from running state to ready state.	In non-pre-empting scheduling a process switches from running state to waiting state.
2.	Also process switches from waiting state to ready state.	In this, a process may get terminate.
3.	Here, multiple processes can run, one process can be pre-empted to run another process.	In this, when the process assigned to CPU, it keeps process busy, till it gets terminate.
4.	It needs specific platform.	It is platform (hardware) independent.
5.	e.g. Windows 95 and MAC.	e.g. Window 3.X versions.

4.3 Scheduling Algorithms

Q. List scheduling algorithms. Explain any two with example. **S-10**

- As we have discussed in the last section, the scheduling algorithms could be preemptive or non-preemptive. The algorithms which we are going to learn are as follows :

1. First Come First Serve.
2. Shortest Job First.
3. Priority scheduling
4. Round Robin Scheduling.

4.3.1 First Come First Served Algorithm (FCFS)

Q. What is FCFS algorithm? Explain with example. **W-08**

- FCFS is non-preemptive algorithm. It is easy & simplest CPU scheduling. The process which requests the CPU first is allocated to the CPU first. The FIFO stands for first in first out.
- When process enters for processing, its PCB is connected with the ready queue.
- The concept of FCFS is very easy to understand from the following example we can calculate the performance of FCFS algorithms. It is poor.

e.g.

Table 4.1

Job	CPU Burst time
1	24
2	3
3	5

- If the job takes place in order of 1, 2, 3 then waiting time for process 2 and 3 is long. We can use Gantt chart to represent the same.

Job1	Job2	Job3

1 24 27 32

- The turnaround time for job1 is 24, job2 is 27 and for job3 it is 30.

- Average turnaround time is (24 + 27 + 32) / 3 = 27.66 of the jobs is suffered as follows then.

Job2	Job3	Job1

0 3 8 32

- Average turnaround time is 3 + 8 + 30/3 = 39/3 = 13.15.
- In FCFS if there is a process which is CPU bound and many other processes having I/O bound, then CPU bound process will hold CPU for its execution. At the same time the I/O bound processes finishes their I/O operations and enters in ready queue for an execution. This moment CPU gets idle state. In this way performance of FCFS is poor. Because CPU and I/O are not effectively utilized in this method.
- Especially in the case of time sharing operating system FCFS is not a good solution for the CPU scheduling. The reason behind this is, in time sharing operating system it is necessary that each user should get CPU share after certain interval. It would be terrific to allow one process to keep the CPU for long time.

4.3.2 Shortest Job First (SJF)

Q. Describe the shortest job first scheduling method. Illustrate with example. **S-09**

Q. Solve the following problem by using following scheduling algorithm. **S-09**

Process	Burst time
P1	5
P2	15
P3	12
P4	25
P5	5

1) SJF 2) FCFS

Q. Explain Shortest Job First (SJF) process algorithm in detail with the help of diagram. **S-10**

Q. The jobs are scheduled for execution as follows. Solve the problem by using

(i) SJF (ii) FCFS

Also find average waiting time using Gantt Chart. **W-10**

Process	Arrival	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- The shortest job first is a different approach to CPU scheduling. Shortest job first (SJF) algorithms associate with each job the length of its next CPU burst.
- Wherever CPU is available, it is assigned to the process or job with smallest next CPU burst if the two jobs have same CPU burst time then FCFS is used to solve the problem.
- The more suitable terminology for shortest job first could be the Shortest-next-CPU-Burst algorithm, because scheduling depends on the length of the next CPU burst of the job or process.

e.g.

Table 4.2

Job	Burst time
1	6
2	8
3	7
4	3

- Using shortest job first scheduling, we would schedule these jobs according to following Gantt chart.

Job4	Job1	Job3	Job2

0 3 9 16 24

- The waiting time for job1=3 ms, job2=16 ms, job3=9 and job4=0. Thus the average waiting time is (3+16+9+0)/4= 7 milliseconds. If we use FCFS method, average waiting time would be 10.25 milliseconds.
- Shortest job first is optimal, because it gives minimum average waiting time for a given set of jobs. e.g. from the above diagram it is clear that it reduces the waiting time. This method is optimal. The waiting time for long processes using this method is ultimately reduced as the small process get execute first.
- Let us consider one more example:

Process	Burst time
P1	5
P2	15
P3	12
P4	25
P5	5

- Solution: Using SJF . Processes can be represented using Gantt chart as follows :

P1	P5	P3	P2	P4

0 5 10 22 37 62

(a) Waiting time for P1=0 ms, P2=22 ms, P3=10 ms, P4=37 ms, P5=10.

(b) Average waiting time is ((0+22+10+37+10)/5)= 27.2 ms.

Solution using FCFS algorithm: Processes can be represented using Gantt chart as follows :

P1	P2	P3	P4	P5

0 5 20 32 57 62

(a) Waiting time for P1=0 ms, P2=5 ms, P3=20 ms, P4=32 ms, P5=57.

(b) Average waiting time is ((0+5+20+32+57)/5)= 35.2 ms.

- Though we have said that SJF is optimal, the major difficulty in SJF is to know the length of next CPU burst or request. For a job scheduling in a batch system, we can use job time limit.

Advantages

1. It is optimal, due to average waiting time of process gets reduce.
2. Due to short process first it's waiting time decreases and waiting time of long job increases ultimately the waiting time decreases.

Disadvantages

1. Knowing the length of next CPU burst is very difficult.
2. The job in a queue lefts for execution, at the same time if new jobs set added into queue then scheduler can not choose newly come job though it is having shorter CPU time burst.
3. The SJF algorithm may be either preemptive or non preemptive.
4. The choice arises when a new process arrives at the ready queue while a previous process is executing. The new process may have a shorter next CPU burst than what is left of the currently executing process.
5. A preemptive SJF algorithm will preempt the currently executing process whereas a non preemptive SJF will allow the currently running process to finish its CPU burst.
6. Preemptive SJF scheduling is sometimes called shortest remaining time first scheduling. The method used to find approximate next CPU Burst time is **exponential average**. As an example consider the following four processes with the length of CPU given in milliseconds.

Table 4.3

Process	Arrival	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- Following figure shows the Gantt chart of the above processes :

P_1	P_2	P_4	P_1	P_3

0 1 5 10 17 26

- P1 started at 0, P2 arrived at 1. Remaining time of P1 is 7 which is larger than process P2. So P1 is preempted and P2 will be scheduled.
- After that again P4 is scheduled as its time is less than remaining time of P1. After this P1 gets its remaining time and then P3 is scheduled.
- If the process arrive at ready queue at the times shown and need the indicated burst time, then resulting preemptive SJF is as shown.

$(10 - 1) + (1 - 1) + (17 - 2) + (5 - 3) / 4 = 26/4 = 6.5$ ms

The waiting for

$P_1 = 10 - 1 = 9$

$P_2 = 1 - 1 = 0$

$P_3 = 17 - 2 = 15$

$P_4 = 5 - 3 = 2$

4.3.2.1 Comparison of FCFS and SJF

Q. Compare the FCFS and SJF scheduling algorithms with respect to following:

(i) Turn around time

(ii) Waiting time

(iii) Through put

W-09

FCFS	SJF
FCFS uses simple approach like first come and first serve.	SJF uses shortest job first approach for scheduling
Turnaround time can be low because of context switches occur only when process termination takes place and no reorganization of the process queue is required, scheduling overhead is minimal.	Turnaround time is more than FCFS.
Waiting time and response time can be low for the same reasons above	Waiting time and response time increase as the process' computational requirements increase.
Throughput can be low, since long processes can take over the CPU	This algorithm is designed for maximum throughput in most scenarios.

4.3.3 Priority Scheduling

Q. Consider the following set of processes with the length of the CPU burst time given in milliseconds.

Process	Burst time	Priority
P11	20	4
P12	6	1
P13	15	2
P21	8	3
P22	12	2
P23	5	4

Find the waiting time for each process using the SJF algorithm. **W-09**

- The SJF algorithm is referred as a special case of general priority algorithm. The simple reason behind this is shortest i.e. smallest job gets first execute by SJF, so its kind of priority given to the smallest jobs or processes.
- A priority is associated with each process and the CPU is allocated to the process with the highest priority. In case if CPU burst time of two processes is same then FCFS algorithm is used for scheduling them.
- The SJF algorithm is a priority algorithm where the priority (p) is inversely proportionate to the next CPU burst time. i.e. highest is the CPU burst time lowest is the priority and vice versa.
- In the priority scheduling the range of priorities can be represented as 0 to7 or 0 to 4095 etc. But in some system low numbering can be given to higher priorities or high numbering is given to low priorities. This forms a big confusion for the user to understand the numbering of priority.
- Here let's see the example of higher priorities to lower numbering.

e.g.

Table 4.4

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

- Using priority scheduling these processes can be represent using Gantt chart.

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

- The average waiting time in the above example is 8.2 milliseconds.
- Let us consider one more example here to find out the waiting time of each process:

Process	Burst time	Priority
P11	20	4
P12	6	1
P13	15	2
P21	8	3
P22	12	2
P23	5	4

Solution :

- The above processes can be represented using Gantt chart as follows: The processes are arranged according to their priority.

P12	P13	P22	P21	P11	P23

0 6 21 33 41 61 66

Waiting time for each process is as follows:

- P11= 41 ms, P12=0 ms, P13=6 ms, P21=33 ms, P22=21 ms and P23=61 ms. (ms means milliseconds)
- Priorities can be internal or external. Internally defined priorities are given by considering the factors like time limits, memory requirements, and number of open files and ratio of average I/O burst to average CPU burst. External priorities are external to an operating system. e.g. such as an importance of process, type of work etc. Priority scheme can be preemptive or non-preemptive.
- When process arrives at a ready queue, its priority is compared with the current process.

- A preemptive priority will prompt the CPU priority of newly arrived process is higher than priority of currently running process.
- A non-preemptive priority scheduling algorithms will simply put new process at the head of ready queue.

Advantages of priority scheduling :

1. Simplicity
2. Reasonable support for priorities.
3. Suitable for applications with varying time and resource requirements.

Disadvantages of Priority scheduling :

1. Indefinite blocking or starvation. A process that is ready to run but lacking CPU can be considered blocked or waiting for CPU.
2. A priority scheduling can leave some low priority waiting processes indefinitely for CPU.
3. If system eventually crashes then all unfinished low priority processes gets lost.

4.3.3.1 Indefinite Blocking or Starvation (Disadvantage of Priority Scheduling)

- This is a disadvantage of priority scheduling called indefinite blocking or starvation. A process that is ready to run but lacking CPU can be considered blocked or waiting for CPU.
- There are more disadvantages of priority scheduling :
  1. A priority scheduling can leave some low priority waiting processes indefinitely for CPU.
  2. If system eventually crashes then all unfinished low priority process gets lost. An aging technique is used to overcome such type of problems. **Aging** is a technique used which gradually increases the priority of process that is waiting in the system for a long time.

4.3.4 The Round Robin Scheduling (RR)

Q. Describe the Round Robin algorithm with suitable example. **S-09**

Q. Explain the Round Robin scheduling algorithm with sample example. **W-09**

Q. Explain Round Robin (RR) scheduling with an example. **W-10**

- The Round Robin (RR) scheduling algorithm is specially used for time sharing systems. It is quite same like FCFS, only difference is that in Round Robin preemption is added to switch between the processes.
- A small unit of time is called as **time quantum** or **time Slice.** A time quantum is generally from 10 to 100 milliseconds. In this method a ready queue is treated as a circular queue. The CPU scheduler goes around this ready queue and allocates CPU to each an every process for a time interval of up to 1 time quantum.
- For implementing this scheduling, the processes are kept in FIFO order. The new processes are added to the tail i.e. The last position of the ready queue. The scheduler picks one job and assigns it to CPU; CPU executes it and bursts for a time quantum. In this case there are two possibilities.
- If time quantum of processor is greater than process burst time then, process itself releases CPU, otherwise interrupt interrupts the CPU. Then CPU stops the execution and the process is shifted to the tail of the ready process queue. After this, CPU scheduler selects next job for execution.
- The average waiting time under RR policy is often quite long.

e.g.

Table 4.5

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 get first 4 milliseconds. Since it requires

another 20 milliseconds, it is preempted after the first time quantum and CPU is given to next process in queue.
- Process P_2 since it does not need 4 ms it quits before its time quantum expires. CPU is then given next process.
- Process P_3 has received one time quantum. Then CPU is returned to process P_1 for an additional time quantum.
- The resulting Round Robin schedule is :

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

- Waiting time for p1=0+6, p2=4 and p3=7
- The average waiting time is (6+4+7)/3
- That is : 17/3=5.66 ms
- The average waiting time is 17/3=5.66 m/s
- In RR scheduling algorithm, no process is allocated the CPU for more than 1 time quantum in a row i.e. every process will get 1 quantum at a time. If a CPU burst exceeds 1 time quantum then it is preempted and is put back in queue. The RR scheduling algorithm is preemptive.
- The performance of Round Robin scheduling is highly dependent on the time quantum which is given for each process to get execute. If the time quantum is largest the RR algorithm is as good as FCFS. In case if time quantum is small then RR algorithm is called as **processor sharing** and this creates an appearance that each process has got its own processor running at 1/n th the speed of the real processor.

Advantage

- The average waiting time is minimal (negligible).

Disadvantage

- If the time quantum is too large then it is as good as FCFS policy.

Process P_2 since it does not need 4 m/s it quits before its time quantum expires. CPU is then given next process.

- Process P_3 has received one time quantum. Then CPU is returned to process P_1 for an additional time quantum.
- The resulting Round Robin schedule is :

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

- The average waiting time is 17/3=5.66 m/s
- In RR scheduling algorithm, no process is allocated the CPU for more than 1 time quantum in a row i.e. every process will get 1 quantum at a time.If a CPU burst exceeds 1 time quantum then it is preempted and is put back in queue. The RR scheduling algorithm is preemptive.
- The performance of Round Robin scheduling is highly dependent on the time quantum which is given for each process to get execute. If the time quantum is largest the RR algorithm is as good as FCFS. In case if time quantum is small then RR algorithm is called as **processor sharing** and this creates an appearance that each process has got its own processor running at 1/n th the speed of the real processor.

Advantage

- The average waiting time is minimal (negligible).

Disadvantage

- If the time quantum is too large then it is as good as FCFS policy.

4.4 Other Scheduling Algorithms

Even some other scheduling methods which we are going to study are as follows:

1. Multilevel
2. Multiprocessor
3. Real-time.

4.4.1 Multilevel Queue Scheduling

- Multilevel queue scheduling is the class of scheduling algorithms which has been created for situations in which jobs or processes are easily classified into different groups. e.g. a common division is made between foreground (interactive) jobs and background (batch) jobs.
- These two types of jobs have quite different response time requirements, and so might have different scheduling algorithms. Also foreground processes may have priority over background jobs.
- The multilevel queue scheduling makes partitions into ready queue into separate queues. Jobs are permanently assigned to one queue. Each queue has its own scheduling algorithm.
- The foreground queue might be scheduled by a round-Robin algorithm, while the background queue is scheduled by FCFS algorithm.
- There must be scheduling among the queues, which is commonly implemented as fixed priority preemptive scheduling. For example, foreground queue may have absolute priority over background queue.
- Lets take an example of multilevel queue scheduling algorithm with five queues listed below in order of priority.
  - System process
  - Interactive processes

- Interactive editing processes
- Batch processes
- Student processes

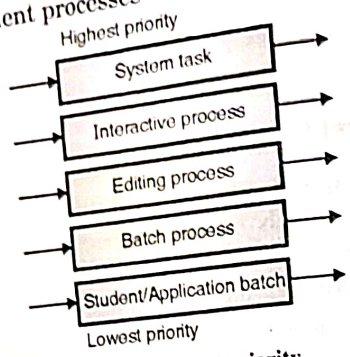

Fig. 4.3 : Lowest priority

- For example there is queue in the batch process cannot get execute till the queue of system processes, interactive processes and interactive editing processes gets execute. If they are empty, then if batch process starts executing then if job queue at editing processes arrives the batch job will be preempted.
- There is one more possibility in this scheduling. As we have separated queues by classifying them into foreground and background processes, the time could be sliced and can be assigned to different queues. After this the individual queues can assign further time slice to individual process. In this case background processes may get 20% time and 80% time will be given to foreground processes.

4.4.2 Multilevel Feedback-queue Scheduling

1. In multilevel queue scheduling once the process is assigned to queue it has to permanently be in that queue. In this queue scheduling there are two types of processes background and foreground. So once queue is

formed either for background queue or foreground queue, process cannot change its queue. Due to this method the overhead lower. But there is one drawback that this method is not flexible.

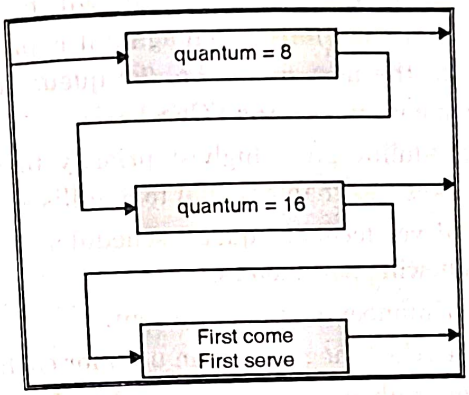

Fig. 4.4 : Multilevel feedback queue scheduling

2. On the other hand in multilevel feedback-queue scheduling algorithm allows the process to move from one queue to other queue. The reason behind this is to separate the processes according to their CPU burst characteristics.
3. If a process uses CPU for long time, it will be shifted to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queue. This type of aging prevents starvation.
4. Consider multilevel feedback queue with three queues from number 0 to 2 processes in queue. In this queue first all processes from 0th queue will get execute, then processes from queue 1 and then queue 2. If process from queue number 0 arrives when queue 1 is getting execute, process from queue 1 will be preempted and process from 0 queue will get execute. Similarly while queue 2 process gets execute, if process in queue 1 arrives then queue 2 process will be preempted and process from queue 1 will be executed.

5. A process which enters in the queue 0 will be given milliseconds. If first process does not finish within milliseconds, then it will be shift to the tail (end) of queue. If this queue is empty, process will be moved the 1st queue and 16 milliseconds will be assigned Still if it does not finish then again it is preempted and shifted to the next queue i.e. 2nd queue. Processes the 2nd queue runs on the FCFS basis.
6. This scheduling gives highest priority to any process which takes less than or equal to 8 milliseconds.
7. In multilevel feedback-queue scheduler is defined using following parameters.
   1. The number of queues present.
   2. The scheduling algorithm used for each queue.
   3. The method or procedure used to determine when to upgrade a process to a higher priority queue.
   4. The method or procedure used to determine when to demote a process to a lower priority queue.
   5. The method or procedure used to determine which queue a process will enter when that process needs service.

4.4.3 Multiple Processors Scheduling (Real-Time)

Q. Explain multi-processor system in detail. **W-08**

- Multilevel queue scheduling deals with a single processor. In case of multiprocessor system it is complicated to schedule process. In this scheduling we use homogeneous (uniform) processors in terms of their functionality. In this type of scheduling we use common ready queues.
- All the processes are getting joined to ready queue and it is assigned to available processor.

- The two approaches are used to schedule the job. In first approach each processor is self-scheduling. CPU itself examines the common ready queue and selects the process to execute. It is known as **symmetric multiprocessing system** i.e **SMP.**
- In other approach two processors cannot choose the same process and process cannot lost from queues. This is to be done by appointing one processor as scheduler for other processors in real time scheduling. It is referred as **asymmetric multiprocessing system.**
- The scheduling facility is required to support real time computing within a general purpose computer system. Real time computing is divided into two types :
  1. Hard real time systems.
  2. Soft real time computing.

4.5 Introduction to Deadlocks

Q. What is deadlock? **W-08**

Q. What is dead-lock? State methods/techniques to handle the deadlock. **W-09**

Q. State the meaning of dead lock. Explain how dead lock can be handed. **S-10**

- Processes compete for physical and logical resources in the multiprogramming environment. A process requests resources, if those are not available at that time, a process enters into the wait state.
- It may happen that waiting processes will never change the state again, because resources requested by the process is occupied by some other process. This is known as deadlock.
- And this condition leads to deadlocks. Deadlocks affect the progress by causing indefinite delays in resource