الملزمة الرابعة

CPU-Scheduling

- CPU-scheduling is the basis of multiprogrammed O.S. by switching the CPU among processes, the O.S can make the computer productive.
- In multiprogrammed O.S, when a process has to wait, O.S takes the CPU away from that process and gives the CPU to another process.

Scheduling Criteria

Many criteria have been suggested for comparing CPU scheduling algorithms, which include the following:

1. *CPU utilization*: we want to keep the CPU as busy as possible.
2. *Throughput*: the number of processes that are computed per time unit.
3. *Turnaround time*: the interval from the time of submission of a process to the time of completion. Turnaround time is the sum of the periods spend waiting to get into memory, waiting in the ready queue, executing on the CPU, doing I/O.
4. *Waiting time*: the sum of the periods spent waiting in the ready queue.
5. *Response time*: is the time that process takes to produce the first response.

Note: it is desirable to maximize CPU utilization and throughput, and to minimize turnaround time, waiting time, and response time.

Scheduling Algorithms

Deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.

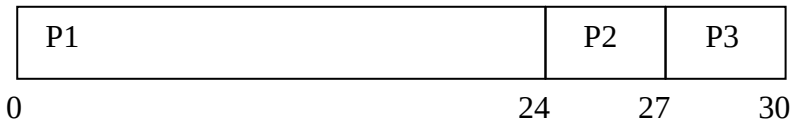
1. First-Come, First-Served algorithm (FCFS)

- the simplest CPU scheduling algorithm. In this scheme the process that requests the CPU first is allocated the CPU first.
- It implements by using FIFO queue.
- The average waiting time (AWT) under FCFS policy, is often quit long.

- Consider the following set of processes that arrive at time 0, with the following length of the CPU-burst time as:

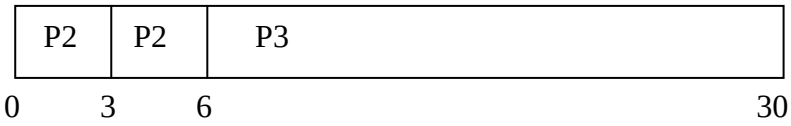
Process	burst-time
P1	24
P2	3
P3	3

- if the processes arrive in the order P1, P2, P3, we get the result by using Gantt chart.



- waiting time for P1=0, P2=24, P3=27
- the AWT=(0+24=27)/3= 17 ms.

- if the processes arrive in the order P2, P3, P1, then:



- the AWT now is (6+0+3)/3= 3 ms.

so, the average waiting time under FCFS policy is generally not minimal, and may vary if the process burst-times vary greatly.

- the FCFS algorithms is *nonpreemptive*. Once the CPU has been allocated to a processor, that process keeps the CPU until it release the CPU, or by I/O request.

2. Shortest-Job-First algorithm (SJF)

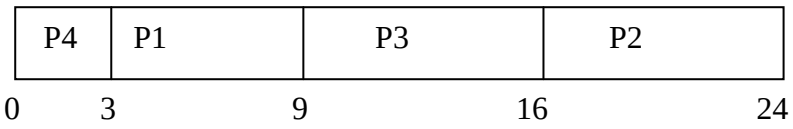
- This algorithm select the process that has the smallest CPU burst.
- If two processes have the same length CPU burst, FCFS algorithm is used to break the tie.
- Two schemes:
  - a- *nonpreemptive* – once CPU given to the process it cannot be preempted until completes its CPU burst

b- *preemptive* – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)

- SJF is *optimal* – gives minimum average waiting time for a given set of processes
- Consider the following set of processes, with the length of CPU-burst time:

Process	burst-time
P1	6
P2	8
P3	7
P4	3

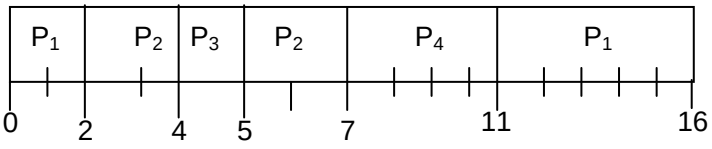
Gantt chart



- The AWT is (0+3+9+16)/4=7 ms.

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4



Average waiting time =((11-2)+(5-4)+(4-4)+(7-5)/4 = (9 + 1 + 0 +2)/4 = 3

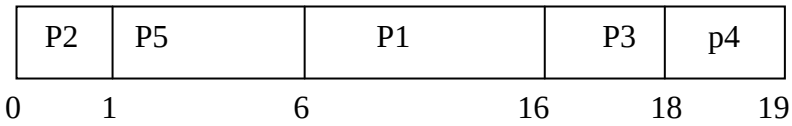
3. Priority Scheduling

- The SJF is special case of the general priority scheduling algorithm.
- Apriority is assigned with each process, and the CPU is allocated to the process with highest priority. Equal priority processes are scheduled in FCFS order.

- Consider the following set of processes, with the length of the CPU-burst time, and priority

Process	burst-time	priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Gantt chart



- the AWT= 8.2 ms.
- Priorities can be either internally (such as: time, memory requirements, ….) or externally (such as: process importance, process department, …).
- A major problem with priority algorithm is *starvation*, which means leaving some low priority processes waiting indefinitely for the CPU.
- A solution to this problem is *aging*. Is a technique of gradually increasing the priority of processes that wait in the system for along time.

4. Round-Robin Algorithm (RR)

- Designed especially for time-sharing systems.
- It is similar to FCFS algorithm, but *preemption* is added to switch between processes.
- 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 called *quantum* (time slice), which is generally from (10-100 ms).
- *Implementation*: keep the ready queue as a FIFO queue of processes

- if the process have a CPU burst of less than one time quantum, then the process will release the CPU, and the scheduler will then proceed to the next process in the ready queue.

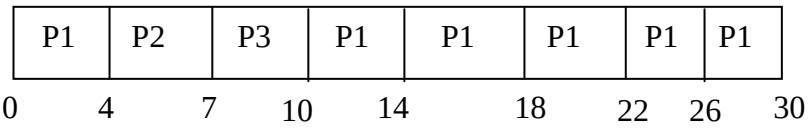
- If the CPU burst of the currently running process is longer than one quantum, the process will be put at the tail of the ready queue, and the CPU scheduler will then select the next process in the ready queue.

- the AWT of RR is long.
- Consider the following set of processes that arrive at time 0:

Process	burst time
P1	24
P2	3
P3	3

And a time quantum Q= 4 ms, so the resulting RR is:

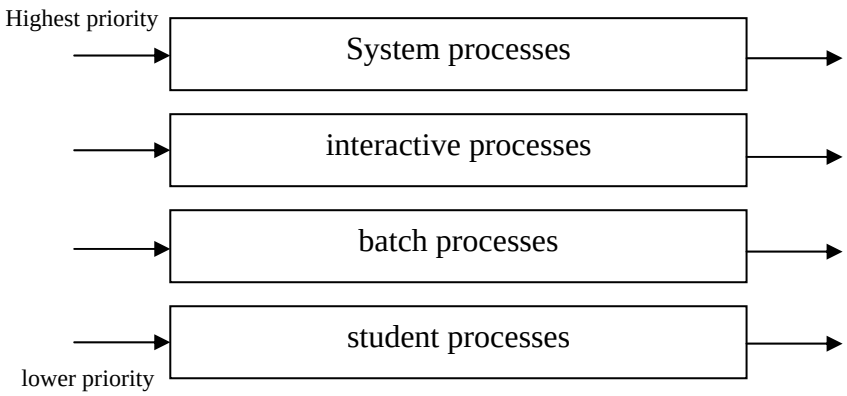
Gantt chart



- the AWT is (4+7+(0+ [10-4])/3= 17/3= 5.66 ms.

5. Multilevel Queue Scheduling

- This algorithm partitions the ready queue into *several separate* queues.
- The processes are *permanently* assigned to one queue on based on some property of the process, such as memory size, process priority,…
- Each process has its own scheduling algorithm, the foreground queue might be scheduled by RR algorithm, while the background queue is scheduled by an FCFS algorithm.

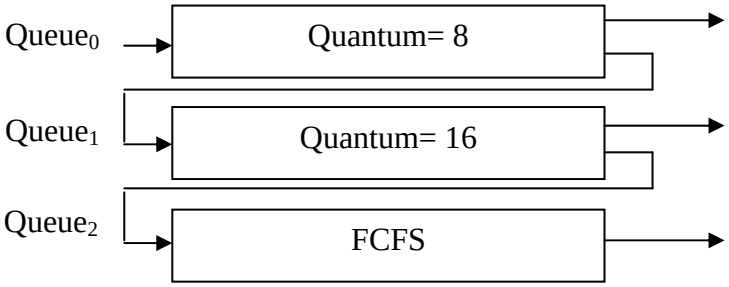


[Multilevel queue scheduling]

- the foreground queue may have the absolute priority over the background queue.

6. Multilevel Feedback-Queue Scheduling

- This algorithm allows a process to *move* between queues.
- If a process uses too much CPU time, it will be moved too a lower-priority queue and a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This prevents starvation.
- Example: consider the following figure:-



- scheduler first executes all processes in $queue_0$.
- Only when queue0 is empty it execute processes in queue1. processes in queue2 will only be execute if (0, 1) are empty.
- A process entering the ready queue is put in $queue_0$ and given a time quantum 8ms. If it does not finish within this time, it is moved to the tail of $queue_1$. if a $queue_0$ is empty, the process at the head of $queue_1$ is given a quantum of 16ms. If it does not complete, it is preempted and is but into $queue_2$. processes in $queue_2$ are run on an FCFS basis, only when queues (0, 1) are empty.

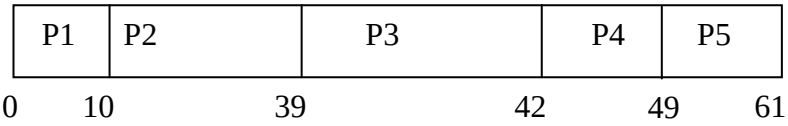
Example: let we have the following set of processes:

Process	burst-time (ms)
P1	10
P2	29
P3	3
P4	7
P5	12

Use FCFS, SJF, RR(Q=10 ms) scheduling algorithm for this set f processes and compute AWT for each algorithm.

1. FCFS

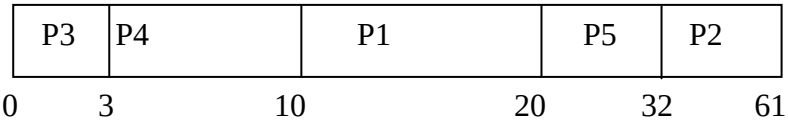
Gantt chart



AWT= (0+10+39+42+49)/5 =28 ms.

2. SJF

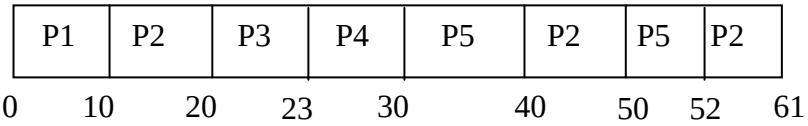
Gantt chart



AWT= (10+32+0+3+20)/5= 13 ms.

3. RR

Gantt chart



AWT=(0+32+20+23+40)/5= 23ms.