SCHEDULING ALGORITHMS

- **Single-Processor Scheduling**
  - ➢ **First-Come, First-Served Scheduling**
  - ➢ **Shortest-Job-First Scheduling**
  - ➢ **Priority Scheduling**
  - ➢ **Round-Robin Scheduling**
  - ➢ **Multilevel Queue Scheduling**
  - ➢ **Multilevel Feedback Queue Scheduling**
- **Multiple-Processor Scheduling**

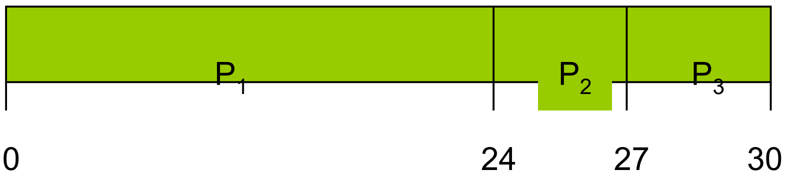
SINGLE-PROCESSOR SCHEDULING

First-Come, First-Served (FCFS) Scheduling

- Simplest CPU scheduling algorithm
- Process that requests the CPU first is allocated the CPU first
- Average waiting time under the FCFS, is often quite long
- Consider the following case:

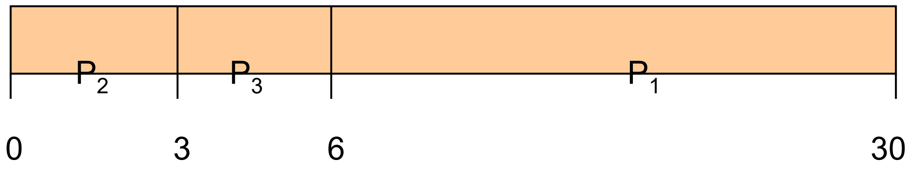
Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
- Then, the Gantt Chart for the schedule is:



✓ Waiting time for P_1=0; P_2=24; P_3 =27
✓ Average waiting time: (0 + 24 + 27) / 3 = 17

- Suppose that the processes arrive in the order: P_2, P_3, P_1
- Then, the Gantt Chart for the schedule is:



✓ Waiting time for P_1=6; P_2=0; P_3=3
✓ Average waiting time: (6 + 0 + 3) / 3 = 3
✓ Much better than previous case
✓ Nonpreemptive
✓ Removes a process from the processor only if it goes into the Wait state or terminates

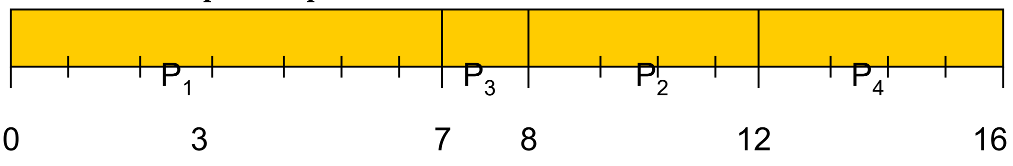
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - Nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst
  - Preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal; gives minimum average waiting time for a given set of processes.

Consider the following case:

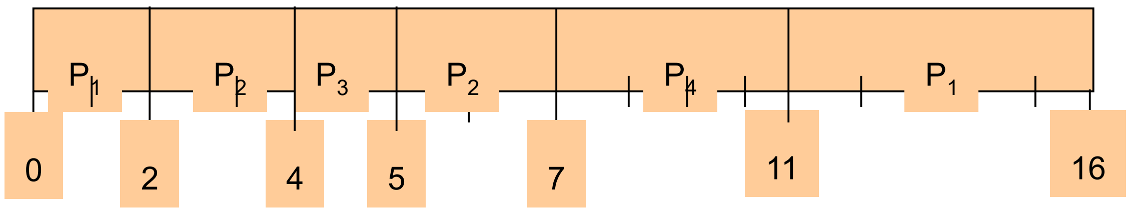
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

➢ Solution in Non-preemptive:



- Average waiting time = (0 + 6 + 3 + 7) / 4 = 4

➢ Solution in Preemptive:



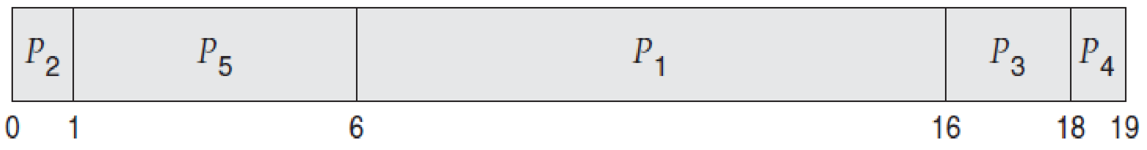
- Average waiting time = (9 + 1 + 0 + 2) / 4 = 3

Priority Scheduling

- A priority number (integer) is associated with each process.
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority). Some systems use low numbers to represent low priority, others use low numbers for high priority
- Priority scheduling can be either preemptive or nonpreemptive.
  - ✓ 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 nonpreemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.
- Equal-priority processes are scheduled in FCFS order.
- SJF algorithm is a priority scheduling; larger the CPU burst, the lower the priority.
- Consider the following case:

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

- The Gantt chart for the schedule is:



  - ✓ Waiting time for P1=6; P2=0; P3=16; P4=18; P5=1
  - ✓ Average waiting time: (6 + 0 + 16 + 18 + 1) / 5 = 8.2

- Priorities can be defined either internally or externally
  - ➢ Internally, some measurable quantity used by OS to compute the priority are:
    - ✓ Time limits
    - ✓ Memory requirements
    - ✓ No.of open files
    - ✓ Ratio of average I/O burst to average CPU burst
  - ➢ Externally, the factors considered are(external to the OS):

- ✓ Importance of the process
- ✓ Type and amount of funds being paid for computer use
- ✓ The department sponsoring the work
- ✓ Political factors

Problem in Priority scheduling algorithm is:

- Indefinite blocking (Starvation)
  - ✓ Process i.e ready to run but lacking the CPU can be considered blocked – waiting for the CPU.
  - ✓ Low priority process waiting indefinitely for the CPU.
  - ✓ In a heavily loaded computer system, a steady stream of higher priority process can prevent a low-priority process from ever getting the CPU.

Solution:

- Aging
  - ✓ Technique of gradually changing the priority of process that wait in the system for a long time.
  - ✓ Eg: Decrement the priority number of a waiting process by 1 every 15 minutes.

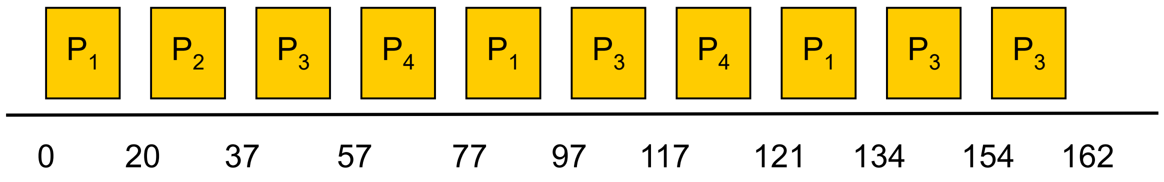
Round Robin (RR)

- Designed especially for time-sharing systems.
- Similar to FCFS.
- Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds.
- After this time has elapsed, the process is preempted and added to the end of the ready queue.
- Ready queue is treated as circular queue
- If the process has a CPU burst of less that 1 time quantum, process itself will release the CPU voluntarily.
- If the CPU burst of the currently running process is longer that 1 time quantum, timer will go off and will cause an interrupt to OS, process will be put at the tail of the ready queue, select the next process. The RR scheduling algorithm is thus Preemptive.
- If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units.
    - Eg: 5 process and a time quantum of 20 milliseconds, each process will get up to 20 milliseconds every 100 milliseconds
- The performance of the RR algorithm depends heavily on the size of the time quantum.
    - If the time quantum is extremely large, the RR policy is the same as the FCFS policy.
    - If the time quantum is extremely small, the RR approach can result in a large number of context switches.

- Consider an example of RR with Time Quantum = 20

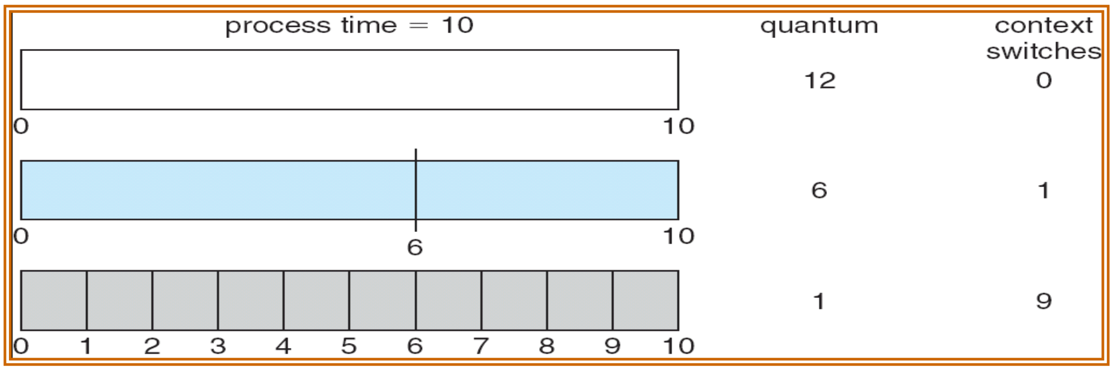
Process	**Burst Time**
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better response

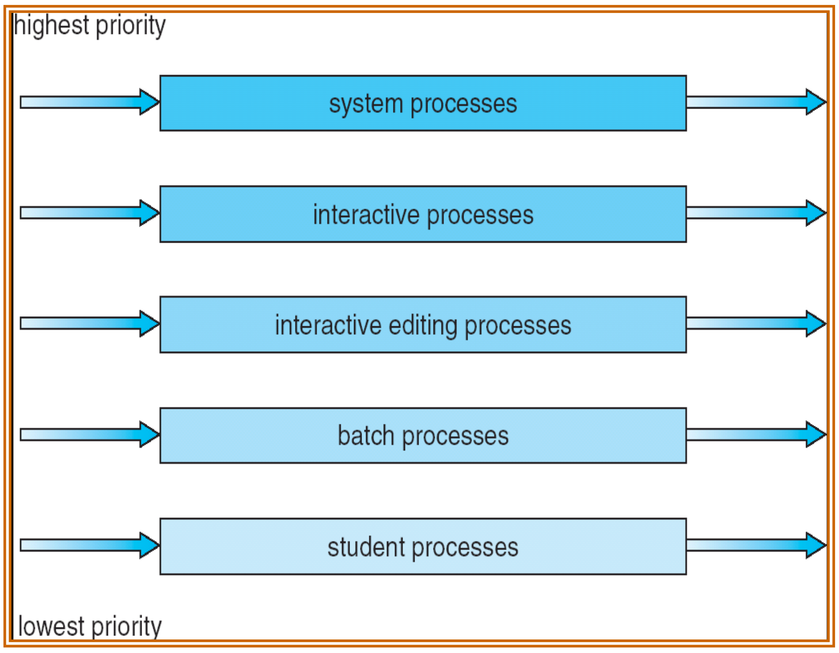
- Time Quantum and Context Switch Time



Multilevel Queue Scheduling

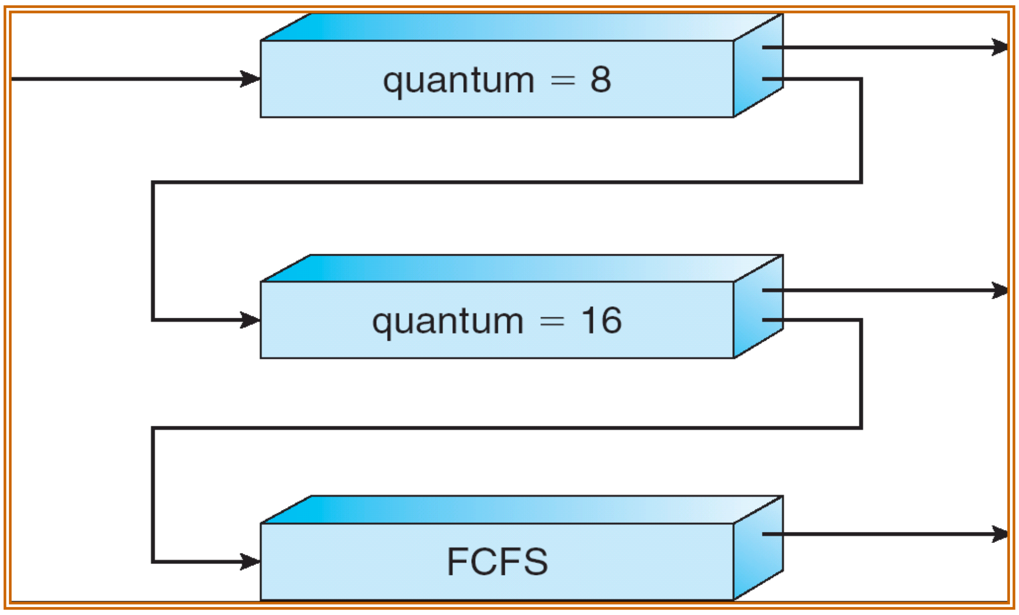
- Ready queue is partitioned into separate queues:
  - Foreground queue for interactive processes.
  - Background queue for batch processes.
- Foreground processes may have higher priority over background processes.
- Processes are permanently assigned to one queue, based on some property such as memory size, process priority or process type.
- Each queue has its own scheduling algorithm. For example,
  - Foreground – RR
  - Background – FCFS
- Scheduling must be done between the queues
  - Fixed priority Preemptive scheduling; (i.e, serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS.

Multilevel Queue Scheduling



Multilevel Feedback Queue Scheduling

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - Number of queues
  - Scheduling algorithms for each queue.
  - Method used to determine when to upgrade a process.
  - Method used to determine when to demote a process.
  - Method used to determine which queue a process will enter when that process needs service.
- Example of Multilevel Feedback Queue:



MULTIPLE-PROCESSOR SCHEDULING

- CPU scheduling more complex when multiple processors are available in a system.
- **Load sharing**
  - If several identical processors are available, then load sharing can occur. It could be possible to provide separate queue for each processor. In this case, however one processor could be idle, with an empty queue, while another processor was very busy.
  - To prevent this situation using a common ready queue, all the process go into one queue and are scheduled onto any available processor. In such a scheme one of two scheduling approaches may be used.

Approaches to Multiple-Processor Scheduling:

1. **Symmetric Multiprocessing (SMP):** In this approach each processor is self scheduling. Each processor examines the common ready queue and selects a process to execute. But it should ensure that two processors do not choose same process.
2. **Asymmetric Multiprocessing:** Appointing one processor as scheduler for the other processors, thus creating a master slave structure.
   - One processor acts as a master server and handles the scheduling decisions, I/O processing and other system activities. The other processors only execute user code.