Chapter 5: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Real-Time Scheduling
- Algorithm Evaluation

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- **CPU–I/O Burst Cycle** – Process execution consists of a *cycle* of CPU execution and I/O wait.

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduler is required to take decision when a process:
  1. Switches from running to waiting state(I/O request)
  2. Switches from running to ready state(Interrupt)
  3. Switches from waiting to ready(I/O completion)
  4. Terminates
- Scheduling under 1 and 4 is *nonpreemptive*.
- All other scheduling is *preemptive.*

- **Non-Preemptive Scheduling** – Once CPU is allocated to a process, the process releases the CPU by its own wish, that is when it no longer requires the CPU.
- Case 1 and 4 of scheduler were non-preemptive.
- **Preemptive Scheduling** – If the CPU can be taken away from a process,the scheduling is said to be preemptive.
- Cases 2 and 3 of scheduler are preemptive.

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running.

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible(maximize)
- **Throughput** – number of processes that complete their execution per time unit(maximize)
- **Turnaround time** – amount of time to execute a particular process(includes time required to wait in the queues,I/O and CPU execution) (minimize)

  Turnaround time is limited by the speed of the device.
- **Waiting time** – amount of time a process has been waiting in the ready queue(minimize)
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced(minimize)

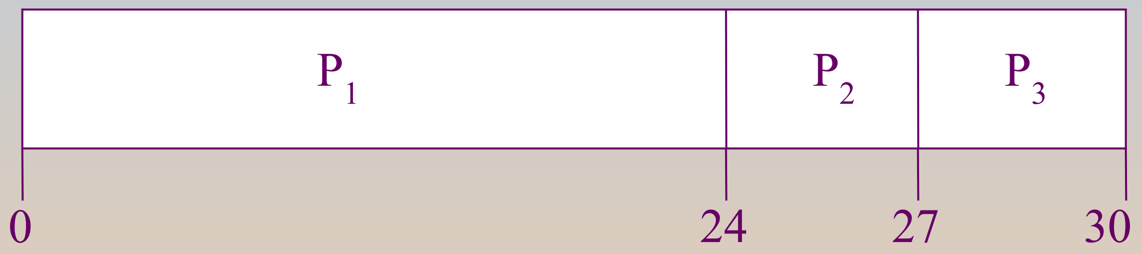
Scheduling Algorithms

- First-come,First-served Scheduling
- Shortest-Job-First Scheduling
- Priority Scheduling
- Round-Robin Scheduling
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling.

First-Come, First-Served (FCFS) Scheduling

Process	CPU 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
  The Gantt Chart for the schedule is:



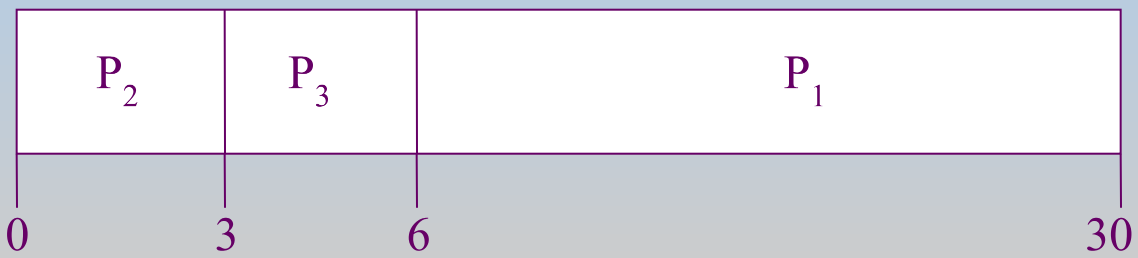
- Waiting time for process $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

P_2 , P_3 , P_1 .

- The Gantt chart for the schedule is:



- Waiting time for process $P_1 = 6; P_2 = 0_; P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case.
- **Convoy Effect** – All processes have to wait for big process to leave the CPU.
- FCFS is non-preemptive.

Shortest-Job-First (SJR) Scheduling

- Two schemes:
  - Non preemptive – 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 know as the Shortest-Remaining-Time-First (SRTF).
- SJF is optimal – gives minimum average waiting time for a given set of processes.

Shortest-Job-First (SJR) Scheduling

- The process having shortest CPU burst is scheduled first.

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

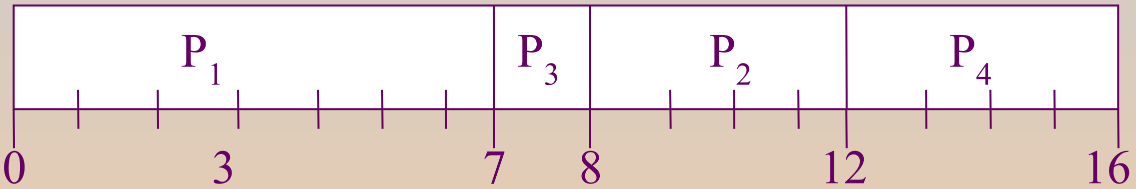
P4	P1	P3	P2
3	9	16	24

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

Example of Non-Preemptive SJF

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

- SJF (non-preemptive)

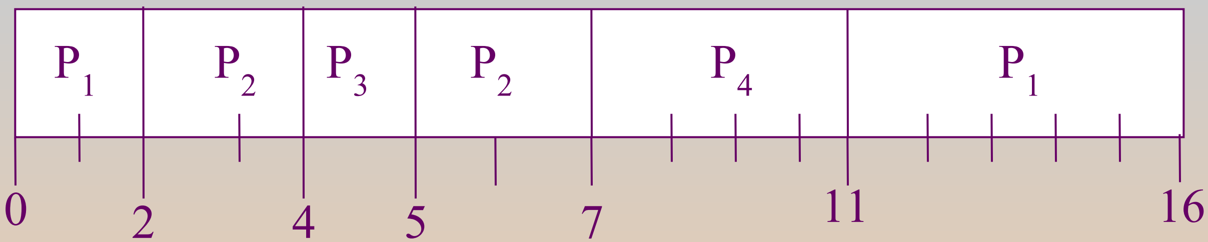


- Average waiting time = (0 +(8-2) + (7-4) + (12-5))/4 = 4

Example of Preemptive SJF

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

- SJF (preemptive)



- Average waiting time = ((11-2) + 1 + 0 +(7-5))/4 = 3
- Preemptive SJF scheduling is also called **shortest-remaining-time-first scheduling.**

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
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Determining Length of Next CPU Burst

- Estimate the length of the next CPU burst of next process.
- This method of approximation is called exponential average.

1. $t_n =$ actual lenght of n^{th} CPU burst
2. $\tau_{n+1} =$ predicted value for the next CPU burst
3. α, $0 \le \alpha \le 1$
4. Define:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n .$$

Priority Scheduling

- A priority is associated with every process.
- The CPU is allocated to the process which has the highest priority (smallest integer ≡ highest priority).
  - Preemptive
  - Nonpreemptive
- SJF is a priority scheduling where priority is the CPU burst time.Larger the CPU burst time, lower the priority.

Priority Scheduling (contd)

- Priority can be defined internally or externally:-
  - **Internal priority** use some measurable quantity like time limit, memory requirements.
  - **External priority** is set up on criteria external to the OS, example- importance of the process, etc.
- Problem with priority scheduling – Indefinite blocking(starvation) may occur for low priority processes.
- Solution to starvation – AGING – Technique of gradually increasing the priority of processes.

  Eg- Priority of every process is decremented every 15minutes.

Priority Scheduling

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

- Priority scheduling

P2	P5	P1	P3	P4
1	6	16	18	19

Average waiting time = (6 + 0 + 16 +18+1)/5 = 8.2

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*). After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*. No process waits more than (n-1)q time units.
- Round Robin scheduling is pre-emptive.

Example of RR with Time Quantum = 4

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- The Gantt chart is:

P_1	P_2	P_3	P1	P_1	P1	P1	P_1		

0 4 7 10 14 18 22 26 30

- Average waiting time = (6+4+7)/3 = 5.6

Round Robin (RR)

- 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.
- Timer interrupts every quantum to schedule next process
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high