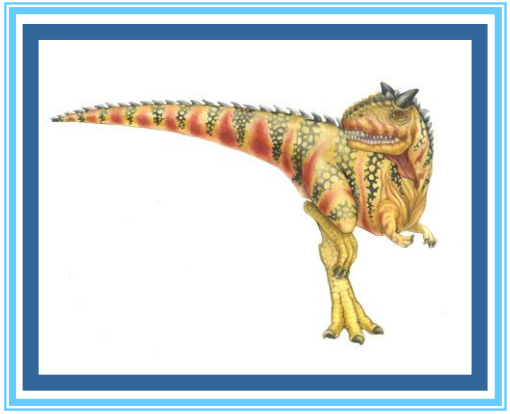
Chapter 6: CPU Scheduling



Chapter 6: CPU Scheduling

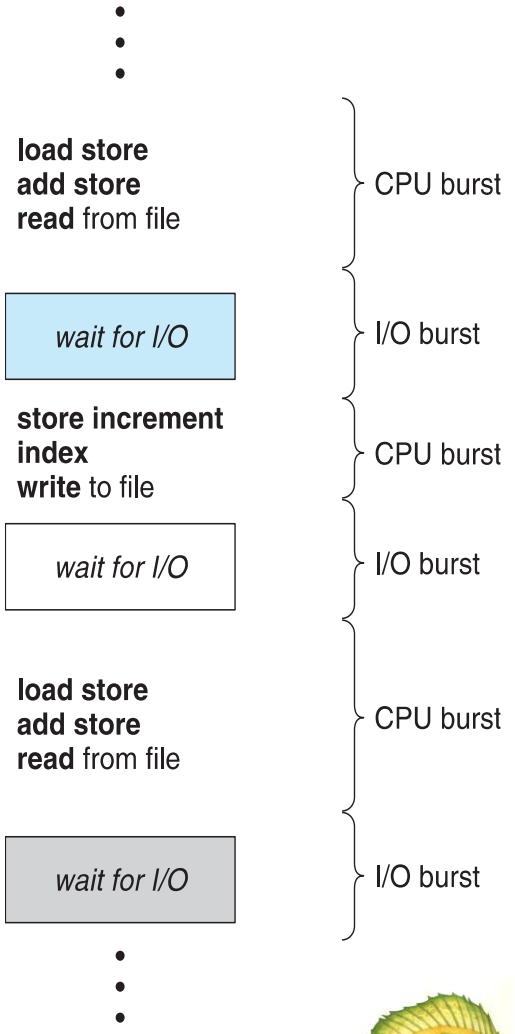
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Thread Scheduling
- Multiple-Processor Scheduling
- Real-Time CPU Scheduling
- Operating Systems Examples
- Algorithm Evaluation

Objectives

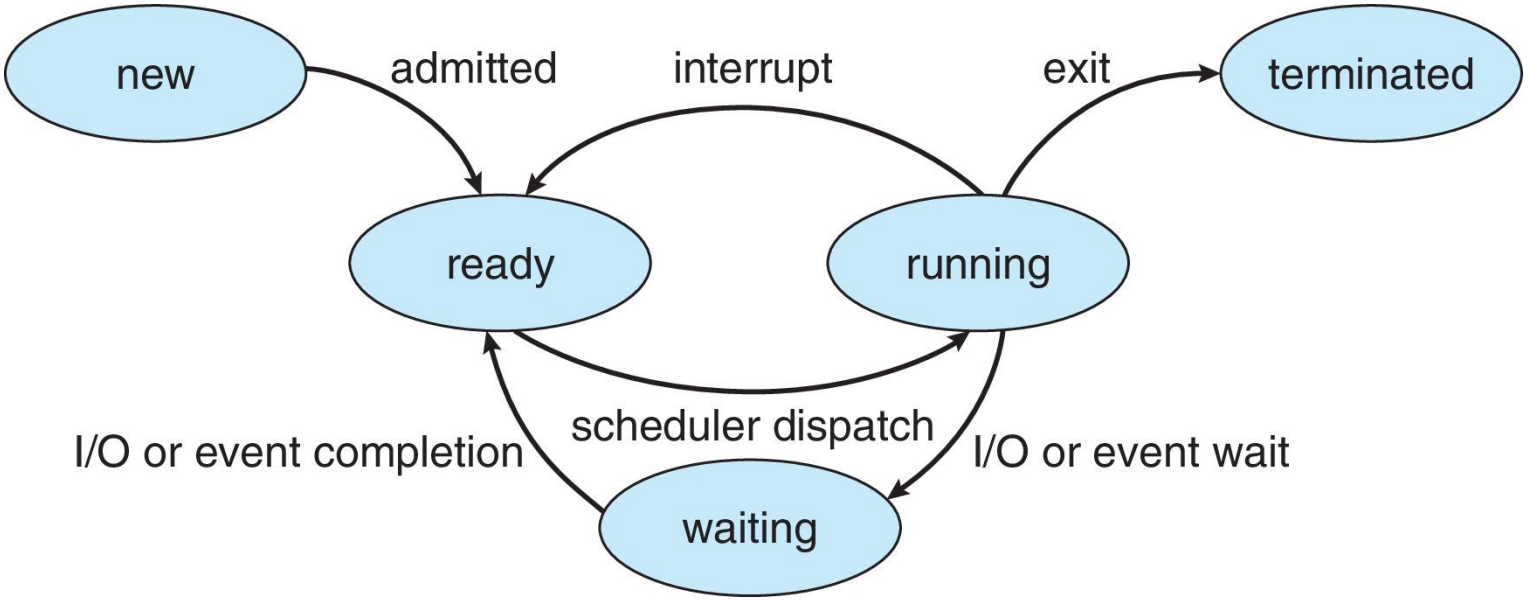
- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system
- To examine the scheduling algorithms of several operating systems

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- Process execution consists of a cycle of CPU execution and I/O wait
- Processes move back & forth between these two states
- A process execution begins with a CPU burst, followed by an I/O burst and then another CPU burst and so on
- CPU–I/O Burst Cycle –
  - Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern



Recall – Process State Diagram



CPU Scheduler

- **Short-term scheduler** selects from among the processes in ready queue, and allocates the CPU to one of them
  - Queue may be ordered in various ways
- 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
- Scheduling under 1 and 4 is **nonpreemptive**
- All other scheduling is **preemptive**

Preemptive vs Non Preemptive

In Pre-emptive scheduling the currently running process may be interrupted and moved to the ready state by OS (forcefully)

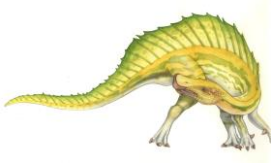
In **Non-preemptive Scheduling**, the running process can only lose the processor voluntarily by terminating or by requesting an I/O. OR, Once CPU given to a process it cannot be preempted until the process completes its CPU burst

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

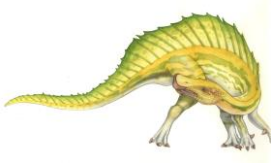
- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)



Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

- Arrival Time
- Burst Time
- Completion Time
- Turnaround Time (Completion time – Arrival time)
- Waiting Time (Turnaround time – Burst time)
- Response Time (CPU allocation Time – Arrival Time



First- Come, First-Served (FCFS) Scheduling

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
  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

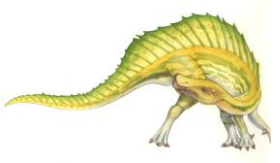
FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

$$P_2 , P_3 , P_1$$

- The Gantt chart for the schedule is:

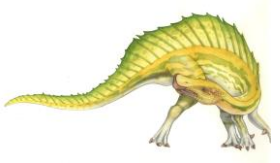
- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- **Convoy effect** - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes



FCFS Scheduling (Cont.)

Example 1:

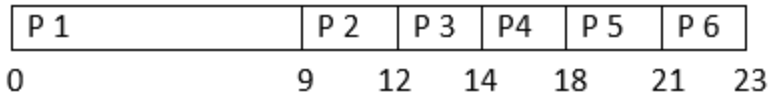
S. No	Process ID	Arrival Time	Burst Time
1	P 1	0	9
2	P 2	1	3
3	P 3	1	2
4	P 4	1	4
5	P 5	2	3
6	P 6	3	2



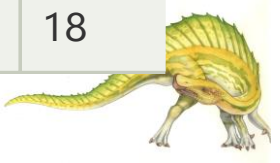
FCFS Scheduling (Cont.)

Example 1: Solution

Gant Chart:



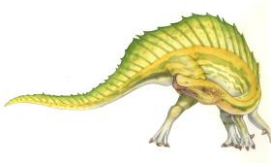
S. No	Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waitir Time
1	P 1	0	9	9	9	0
2	P 2	1	3	12	11	8
3	P 3	1	2	14	13	11
4	P 4	1	4	18	17	13
5	P 5	2	3	21	19	16
6	P 6	3	2	23	20	18



FCFS Scheduling (Cont.)

Example 2:

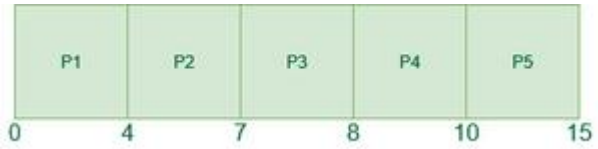
Processes	Arrival Time	Burst Time
P1	0	4
P2	1	3
P3	2	1
P4	3	2
P5	4	5



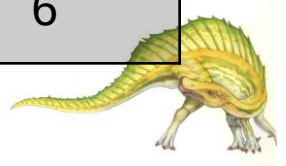
FCFS Scheduling (Cont.)

Example 2: Solution

Gant Chart:



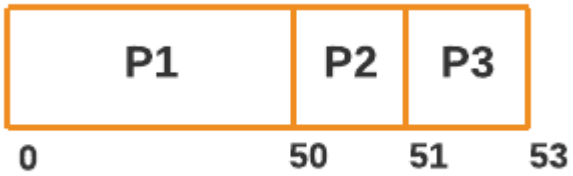
Processes	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	4	4	4	0
P2	1	3	7	6	3
P3	2	1	8	6	5
P4	3	2	10	7	5
P5	4	5	15	11	6



FCFS Scheduling (Cont.)

Example 3: Convey Effect

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	50	50	50	0
P2	1	1	51	50	49
P3	1	2	53	52	50



Average waiting time,
Average WT = (0 + 49 + 50)/3 = 33

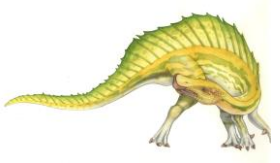
Average waiting Time = (2 + 0 + 1)/3 = 1

FCFS Scheduling (Cont.)

Example 4:

S. No	Process ID	Arrival Time	Burst Time
1	P 1	0	2
2	P 2	1	2
3	P 3	5	3
4	P 4	6	4

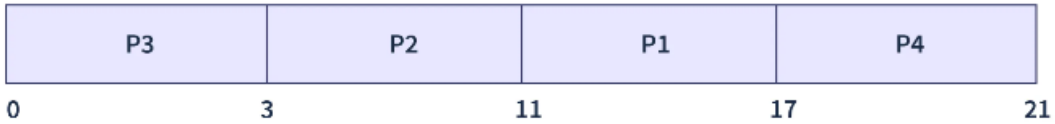
P1		P2	IDLE	P3	P4
0	2	4	5	8	12



FCFS Scheduling (Cont.)

Example 5:

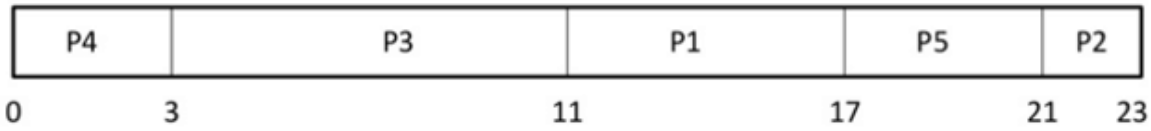
Process	Arrival time	Burst Time
P1	2	6
P2	1	8
P3	0	3
P4	4	4



FCFS Scheduling (Cont.)

Example 6:

Process	Arrival time	Burst time
P1	2	6
P2	5	2
P3	1	8
P4	0	3
P5	4	4



First Come First Serve (cont…)

- ~~Simplest CPU scheduling algorithm~~
- Non preemptive
- The process that requests the CPU first is allocated the CPU first
- Implemented with a FIFO queue. When a process enters the Ready Queue, its PCB is linked on to the tail of the Queue. When the CPU is free it is allocated to the process at the head of the Queue
- **Limitations**
  - FCFS favor long processes as compared to short ones. (Convoy effect)
  - FCFS tends to favor processor bound processes over I/O bound processes
  - Average waiting time is often quite long
  - FCFS is non-preemptive, so it is trouble some for time sharing systems

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

SJF & SRTF Scheduling…

- When the CPU is available it is assigned to the process that has the smallest next CPU burst
- If two processes have the same length next CPU bursts, FCFS scheduling is used to break the tie

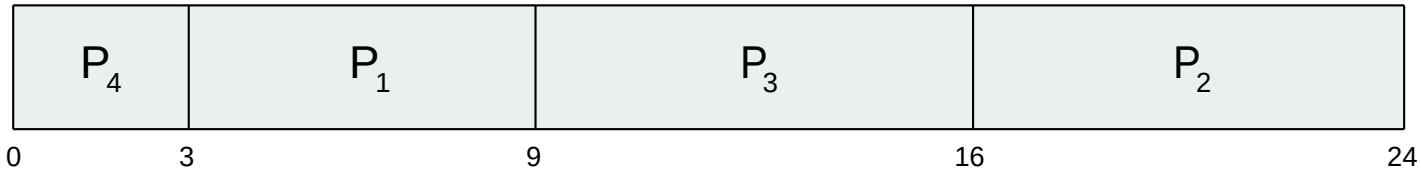
Comes in two flavors

- **Shortest Job First (SJF)**
  - It's a non preemptive algorithm. When a new process arrives having a shorter next CPU burst than what is left of the currently executing process, it ***allows*** the currently running process to finish its CPU burst
- **Shortest Remaining Time First (SRTF)**
  - It's a Preemptive algorithm. When a new process arrives having a shorter next CPU burst than what is left of the currently executing process, it ***preempts*** the currently running process

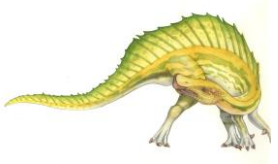
Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



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



Example of SJF

Example 4:

Process	Burst Time	Arrival Time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4