Chapter 6: CPU Scheduling

Chapter 6: CPU Scheduling

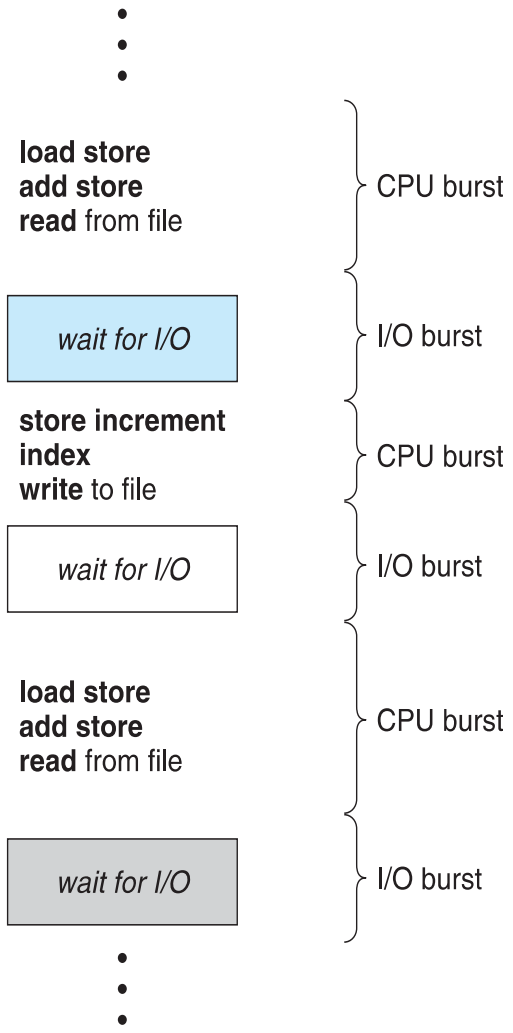
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Thread Scheduling
- Multiple-Processor Scheduling
- Real-Time CPU Scheduling
- Operating Systems Examples
- Algorithm Evaluation

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
- 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



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**
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities

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

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:

$$P_2 , P_3 , P_1$$

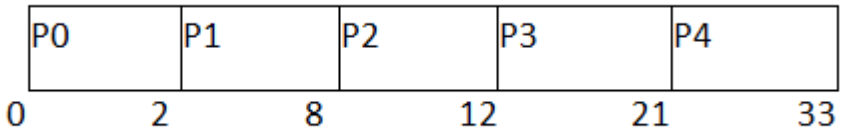
- The Gantt chart for the schedule is:

- 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

FCFS Scheduling (Cont.)

Process ID	Arrival Time	Burst Time	Turn Around Time	Waiting Time
0	0	2	2	0
1	1	6	7	1
2	2	4	10	6
3	3	9	18	9
4	4	12	29	17

Avg Waiting Time=33/5



FCFS Scheduling (Cont.)

- Advantages of FCFS
  - Simple
  - Easy to implement
- Disadvantages of FCFS
  - The scheduling method is non preemptive, the process will run to the completion
  - Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
  - Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

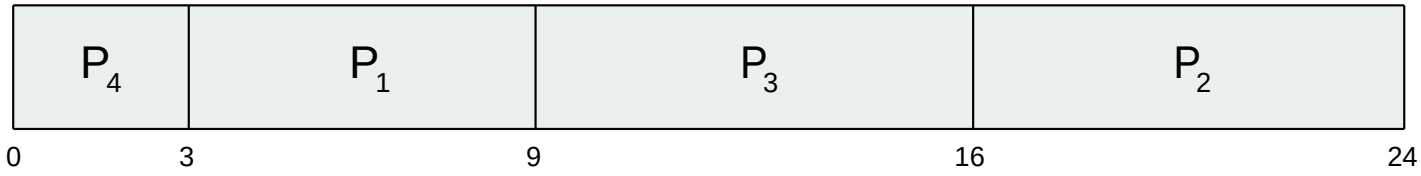
Shortest-Job-First (SJF) Scheduling

- 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

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

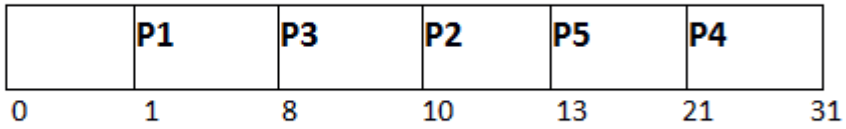
- SJF scheduling chart



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

SJF (cont.)

PID	Arrival Time	Burst Time	Turn Around Time	Waiting Time
1	1	7	7	0
2	3	3	10	7
3	6	2	4	2
4	7	10	24	14
5	9	8	12	4



Avg Waiting Time = 27/5

Determining Length of Next CPU Burst

- SJF algorithm is one of the best scheduling algorithms since it provides the maximum throughput and minimal waiting time
- But The CPU burst time can't be known in advance
- There are the following techniques used for the assumption of CPU burst time for a process.
  1. Static Techniques
  2. Dynamic Techniques
     1. Simple Averaging
     2. Exponential Averaging or Aging

Exponential Averaging or Aging

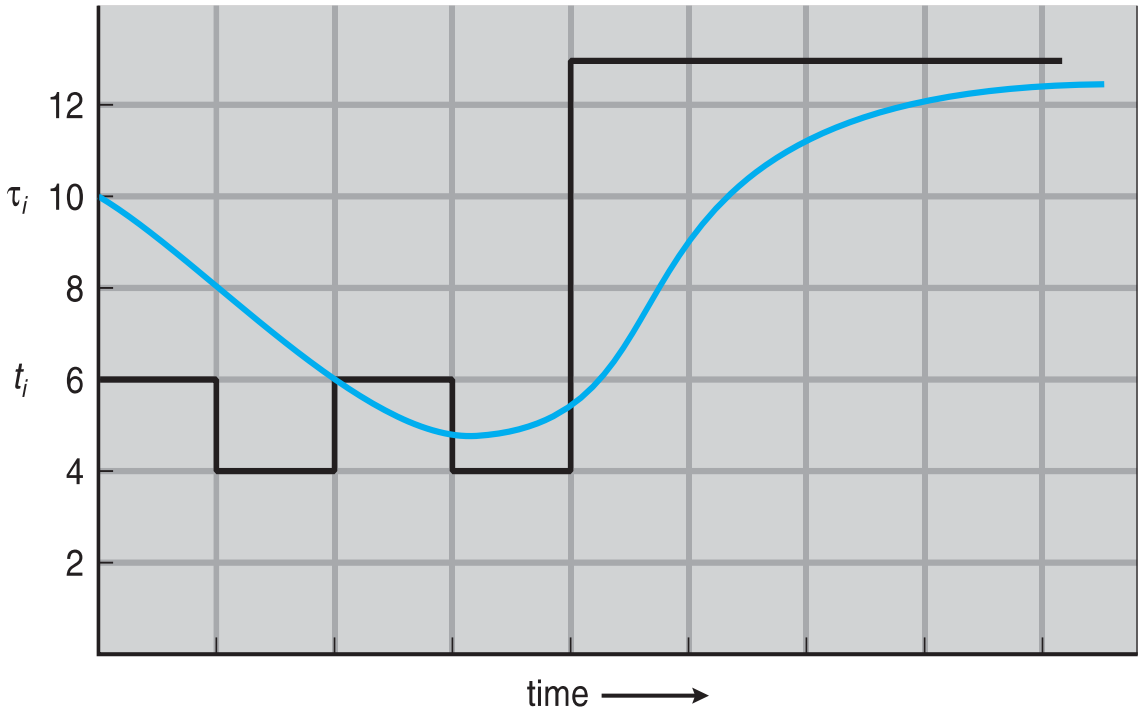
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

- Can be done by using the length of previous CPU bursts, using exponential averaging

  1. t_n = actual length of n^{th} CPU burst
  2. τ_{n+1} = predicted value for the next CPU burst
  3. $\alpha, 0 \le \alpha \le 1$
  4. Define: $\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n$.

- Commonly, α set to ½

Prediction of the Length of the Next CPU Burst



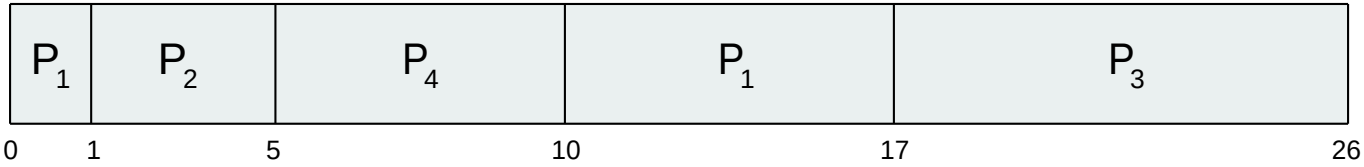
CPU burst (t_i)		6	4	6	4	13	13	13 ...
"guess" (τ_i)	10	8	6	6	5	9	11	12 ...

Example of Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis

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

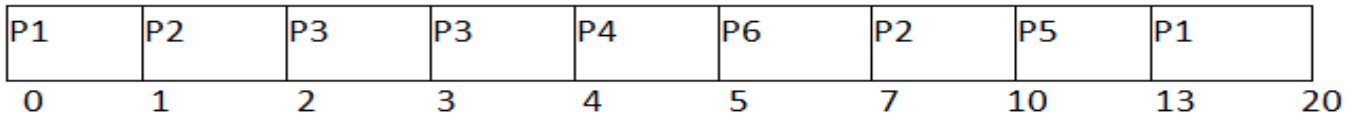
- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

SRJF (cont)

Process ID	Arrival Time	Burst Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	12	0
2	1	4	9	5	0
3	2	2	2	0	0
4	3	1	2	1	1
5	4	3	9	6	6
6	5	2	2	0	0



Avg Waiting Time = 24/6

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)
  - Preemptive
  - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Example of Priority Scheduling

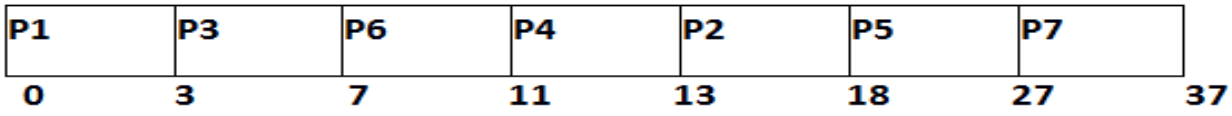
Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart

- Average waiting time = 8.2 msec

Non Preemptive Priority

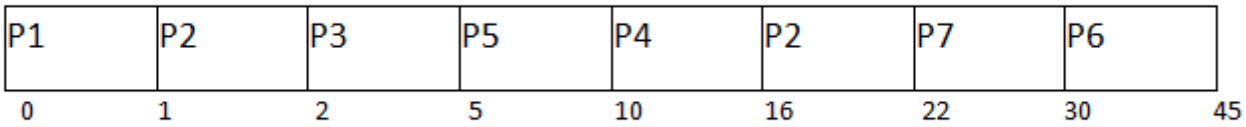
Process Id	Priority	Arrival Time	Burst Time	Turnaround Time	Waiting Time
1	2	0	3	3	0
2	6	2	5	16	11
3	3	1	4	6	2
4	5	4	2	9	7
5	7	6	9	21	12
6	4	5	4	6	2
7	10	7	10	30	18



Avg Waiting Time = (0+11+2+7+12+2+18)/7 = 52/7 units

Preemptive Priority

Process Id	Priority	Arrival Time	Burst Time	Turn around Time	Waiting Time
1	2	0	1	1	0
2	6	1	7	21	14
3	3	2	3	3	0
4	5	3	6	13	7
5	4	4	5	6	1
6	10	5	15	40	25
7	9	6	8	24	16



Avg Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units

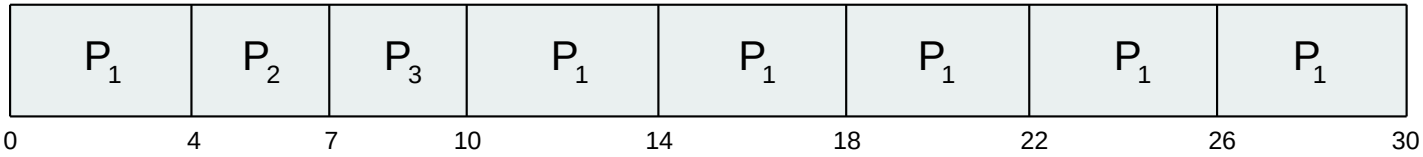
Round Robin (RR)

- Each process gets a small unit of CPU time (**time quantum** *q*), usually 10-100 milliseconds. 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*, 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 $\Rightarrow$ FIFO
  - *q* small $\Rightarrow$ *q* must be large with respect to context switch, otherwise overhead is too high

Example of RR with Time Quantum = 4

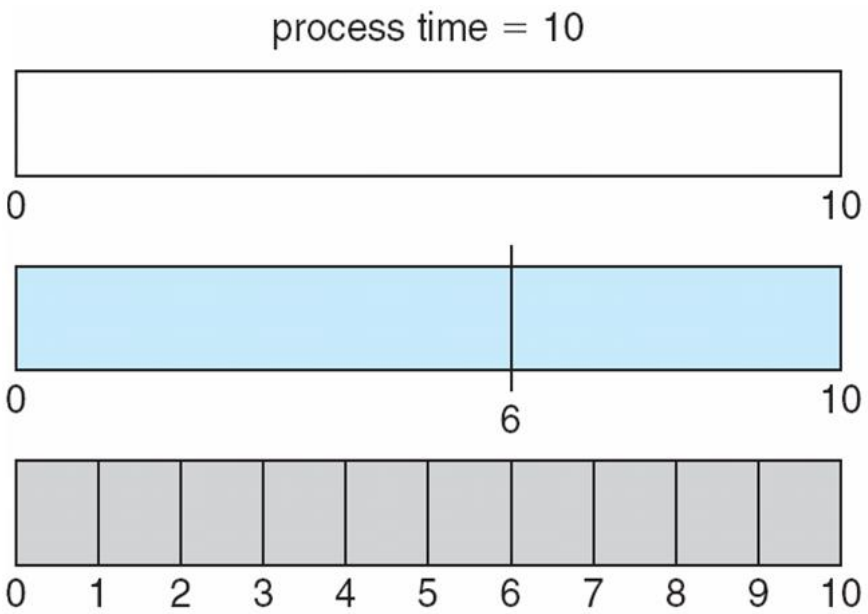
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



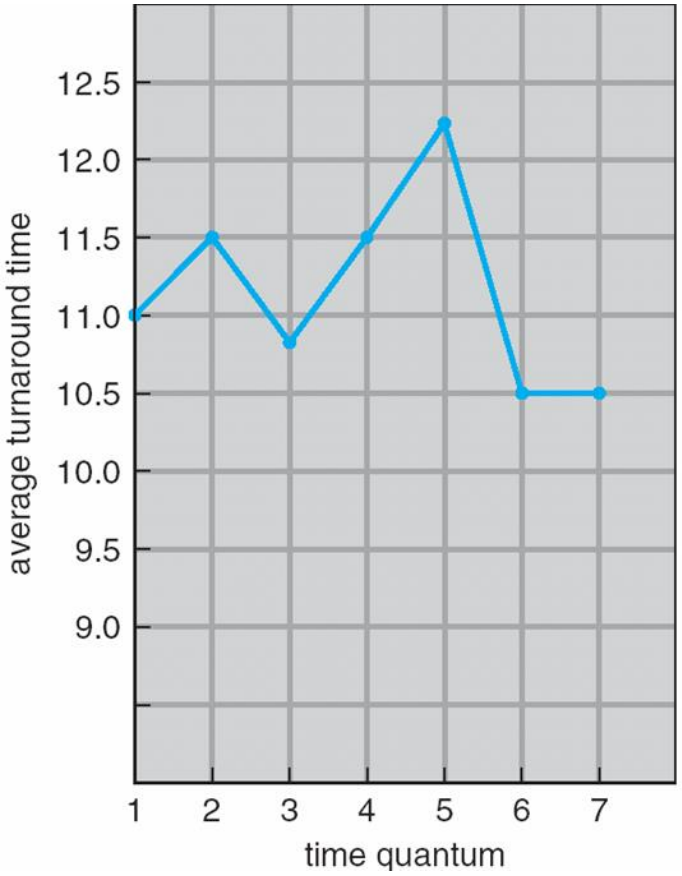
- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% of CPU bursts should be shorter than q

End of Chapter 6