



北京邮电大学软件学院

School Of software Engineering Of BUPT

厚德 博学 敬业 乐群



Operating Systems

Lecture 6 CPU Scheduling

Jinpengchen

Email: jpchen@bupt.edu.cn



Catalog Description

- ⊕ Basic Concepts
- ⊕ Scheduling Criteria
- ⊕ Scheduling Algorithms
- ⊕ Algorithm Evaluation



Chapter 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.



Basic Concepts

- ❁ Scheduling is a fundamental OS function.
 - ❁ Almost all computer resources are scheduled before use.
 - ❁ CPU scheduling is the basis of multiprogrammed OSes.
- ❁ Objective of multiprogramming
 - ❁ Maximum CPU utilization



Basic Concepts

❖ CPU-I/O Burst Cycle

❖ A property of process :

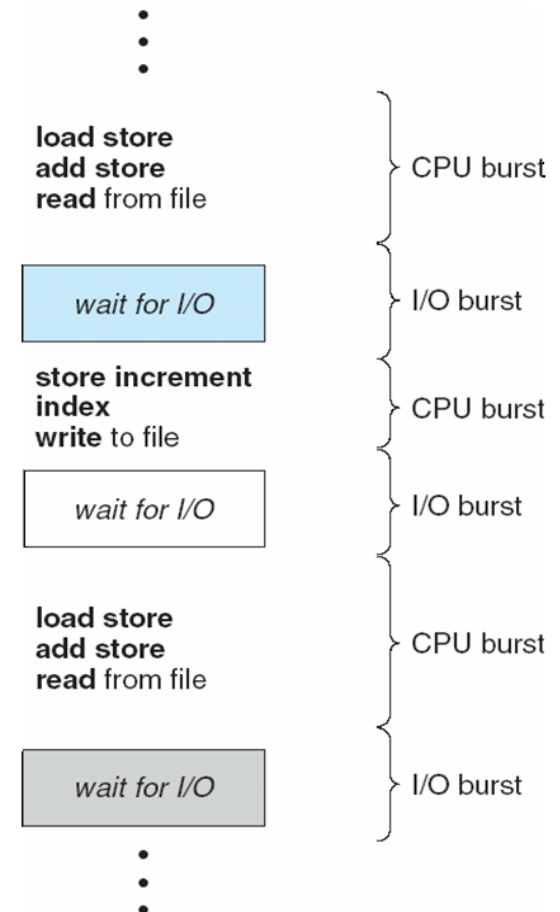
CPU-I/O Burst Cycle

❖ Process execution consists of a cycle of CPU execution and I/O wait

❖ Alternating Sequence of CPU and I/O Bursts

❖ Begin and end with a CPU burst

❖ Process execution
= n (CPU execution +
I/O wait)
+ CPU execution





Basic Concepts

✚ CPU-I/O Burst Cycle

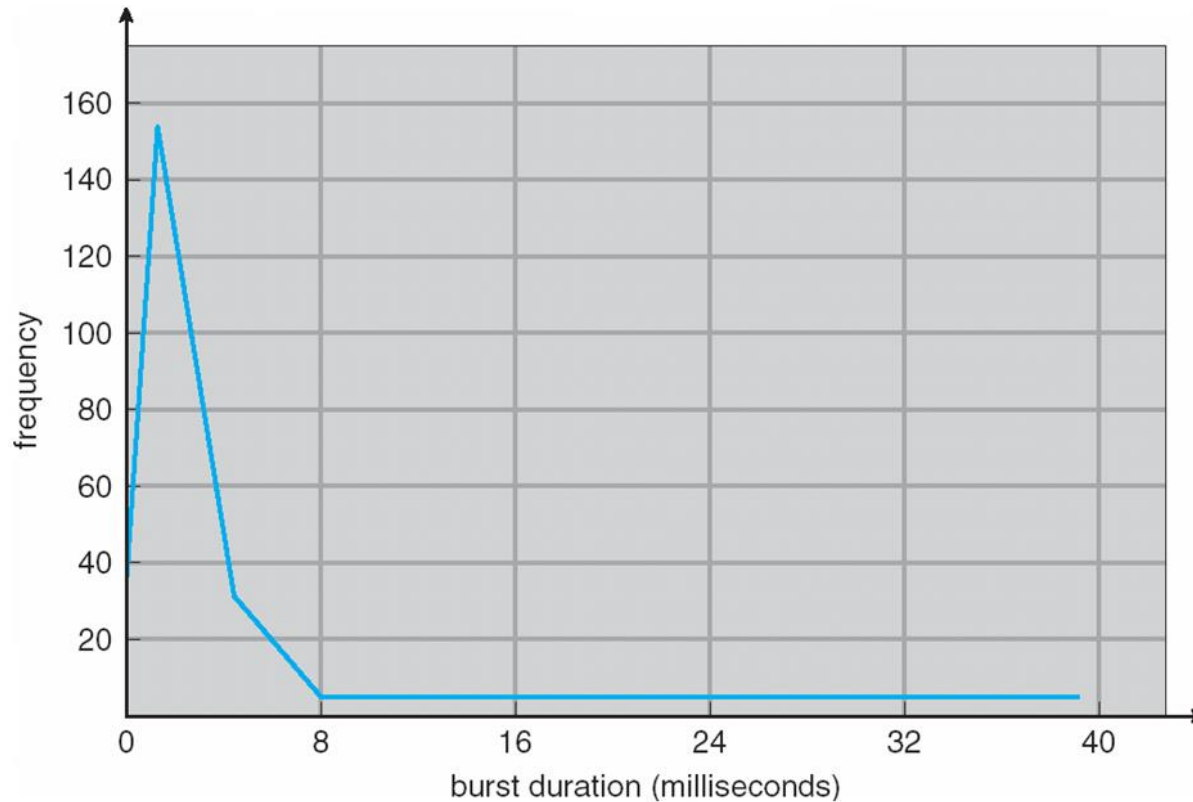
✚ CPU burst distribution

- ✓ CPU burst curve - exponential or hyper exponential
- ✓ CPU-bound program
- ✓ I/O-bound program



Basic Concepts

✚ Histogram of CPU-burst Times





Basic Concepts

❖ CPU Scheduler (Short-term Scheduler)

- ❖ selects a process from the processes in memory that are ready to execute and **allocates** the CPU to the process
- ❖ CPU scheduling decisions may take place when a process:
 - ✓ Switches from **running** to **waiting** state
 - ✓ Switches from **running** to **ready** state
 - ✓ Switches from **waiting** to **ready**
 - ✓ **Terminates**
- ❖ Scheduling under 1 and 4 is **nonpreemptive**
- ❖ All other scheduling is **preemptive**



Basic Concepts

❁ 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
 - ✓ SHOULD be as fast as possible



Catalog Description

- ⊕ Basic Concepts
- ⊕ Scheduling Criteria
- ⊕ Scheduling Algorithms
- ⊕ Algorithm Evaluation



Scheduling Criteria

- ⊕ CPU utilization (CPU 利用率)
 - keep the CPU as busy as possible
- ⊕ Throughput (吞吐率)
- ⊕ Turnaround time (周转时间)
- ⊕ Waiting time (等待时间)
- ⊕ Response time (响应时间)



Scheduling Criteria

⦿ Throughput

- ⦿ # of processes that complete their execution per time unit
 - ✓ different from one process set to another process set
 - ✓ for long processes: may be 1 process per hour
 - ✓ for short transactions: may be 10 processes per second



Scheduling Criteria

✚ Turnaround time

✚ amount of time to execute a particular process

✓ from the time of submission of a process to the time of completion

✓ = the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.



Scheduling Criteria

⊕ 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



Catalog Description

- ⊕ Basic Concepts
- ⊕ Scheduling Criteria
- ⊕ Scheduling Algorithms
- ⊕ Algorithm Evaluation



Scheduling Algorithms

- ✿ FCFS (先来先服务) Scheduling
- ✿ SJF (短作业优先) Scheduling
- ✿ Priority Scheduling (优先级)
- ✿ Round Robin (时间片轮转) Scheduling
- ✿ Multilevel Queue (多级队列) Scheduling
- ✿ Multilevel Feedback Queue (多级反馈队列) Scheduling



Scheduling Algorithms

✚ FCFS (先来先服务) Scheduling

- ✚ Non preemptive (非抢占)

- ✚ Implementation

 - ✓ Normal Queue: FIFO Queue

 - ✓ ordered by request time

 - ✓ linked list

 - ✓ Insert: linked to the tail of the queue

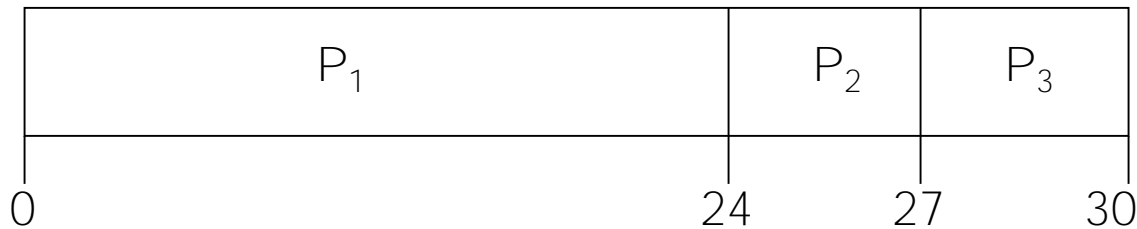
 - ✓ scheduling: removed from the head of the queue



Scheduling Algorithms

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

- Suppose that the processes arrive in the order:
P1 , P2 , P3
The Gantt Chart for the schedule is:



- Waiting time for P1 = 0; P2 = 24; P3 = 27
- Average waiting time: $(0 + 24 + 27)/3 = 17$



Scheduling Algorithms

Suppose that the processes arrive in the order:
P2, P3, P1

The Gantt Chart for the schedule is:



- ⊕ Waiting time for P1 = 6; P2 = 0; P3 = 3
- ⊕ Average waiting time: $(6 + 0 + 3)/3 = 3$
- ⊕ Much better than previous case
- ⊕ Convoy effect short process behind long process



Scheduling Algorithms

- ✿ Convoy effect (护航效应; 护卫效应)
 - ✦ all the other processes wait for the one big process to get off the CPU
 - ✦ \equiv short process behind long process
- ✿ example situation:
 - ✦ one CPU-bound process
 - ✦ many I/O-bound processes



Scheduling Algorithms

✿ Shortest-Job-First (SJF) Scheduling

- ✦ Associate with each process the length of its next CPU burst.
- ✦ Schedule the process with the shortest time.

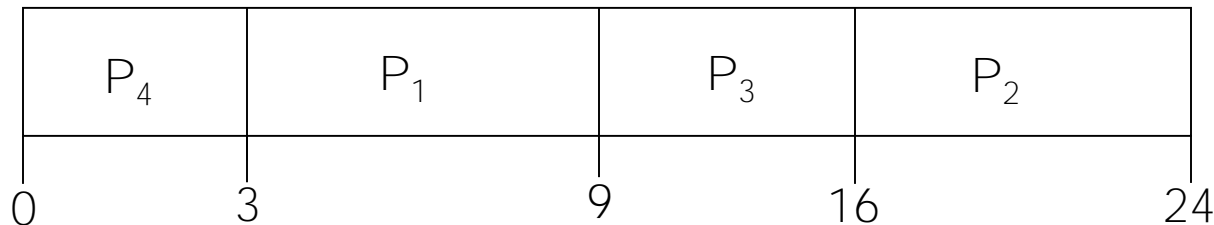


Scheduling Algorithms

❁ SJF scheduling example

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

❁ The Gantt chart for the schedule is:



❁ Waiting time for P1= 3; P2 = 16; P3 = 9; P4 = 0

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

❁ If FCFS, average waiting time: $(0 + 6 + 14 + 21)/4 = 10.25$



Scheduling Algorithms

Shortest-Job-First (SJF) Scheduling

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(最优的)

- gives minimum average waiting time(最小平均等待时间) for a given set of processes

✓ The difficulty is knowing the length of the next CPU request

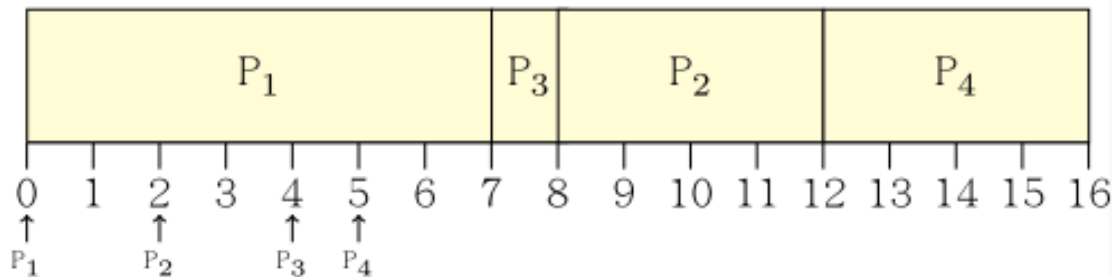


Scheduling Algorithms

Example of Non-Preemptive SJF

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

The Gantt chart for SJF (non-preemptive)



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

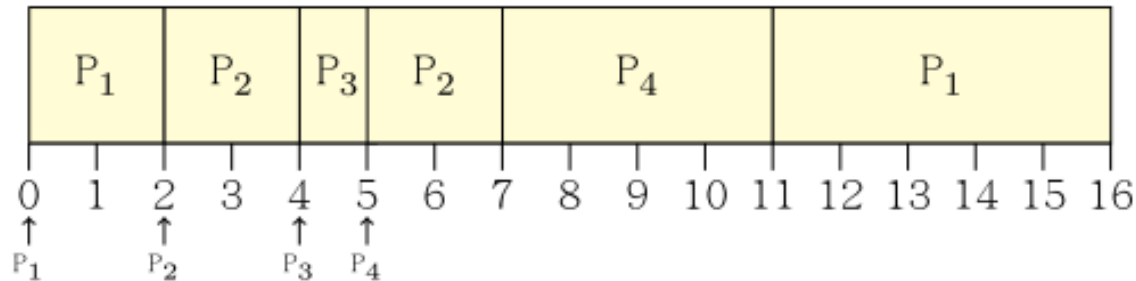


Scheduling Algorithms

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

The Gantt chart for SJF (preemptive)



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



Scheduling Algorithms

❁ Determining Length of Next CPU Burst

- ❁ can only estimate the length
- ❁ can be done by using the length of previous CPU bursts, using exponential averaging (指数平均)
 - ✓ t_n = actual length of nth CPU burst
 - ✓ τ_{n+1} = predicted value for the next CPU burst
 - ✓ Only the actual last CPU burst counts
 - ✓ α , $0 \leq \alpha \leq 1$
 - ✓ Define: $\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n$



Scheduling Algorithms

Examples of Exponential Averaging

■ $\alpha = 0$

- $\tau_{n+1} = \tau_n$
- Recent history does not count

■ $\alpha = 1$

- $\tau_{n+1} = \alpha t_n$
- Only the actual last CPU burst counts

■ If we expand the formula, we get:

$$\begin{aligned}\tau_{n+1} = & \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \dots \\ & + (1 - \alpha)^j \alpha t_{n-j} + \dots \\ & + (1 - \alpha)^{n+1} \tau_0\end{aligned}$$

- ### ■ Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

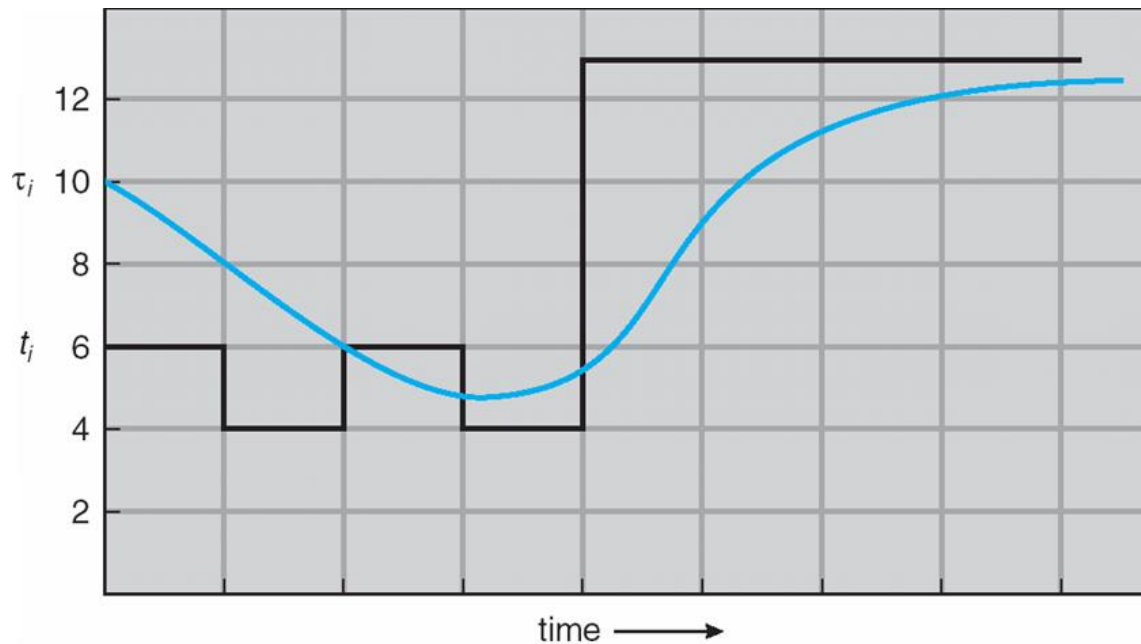




Scheduling Algorithms

● Prediction of the Length of the Next CPU Burst

● Example: $\alpha = 1/2$, $\tau_0 = 10$



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



Scheduling Algorithms

✚ Priority Scheduling

- ✚ A priority number(优先数) is associated with each process
- ✚ The CPU is allocated to the process with the highest priority
- ✚ Priority scheduling can be:
 - ✓ Preemptive VS. Nonpreemptive
- ✚ SJF is a special case of general priority scheduling where priority is the predicted next CPU burst time
- ✚ Problem – Starvation – low priority processes may never execute
- ✚ Solution – Aging – as time progresses, the priority of the process can be increased



Scheduling Algorithms

Round Robin (时间片轮转, RR) Scheduling

- 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.
- 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.
- Performance
 - ✓ q large – FCFS
 - ✓ q small – q must be large with respect to context switch, otherwise **overhead** is too high

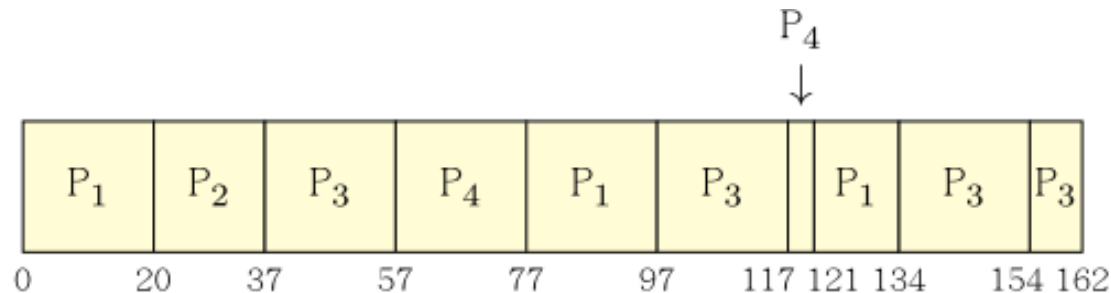


Scheduling Algorithms

Example of RR with Time Quantum = 20

Process	Burst Time
P1	53
P2	17
P3	68
P4	24

The Gantt chart is:



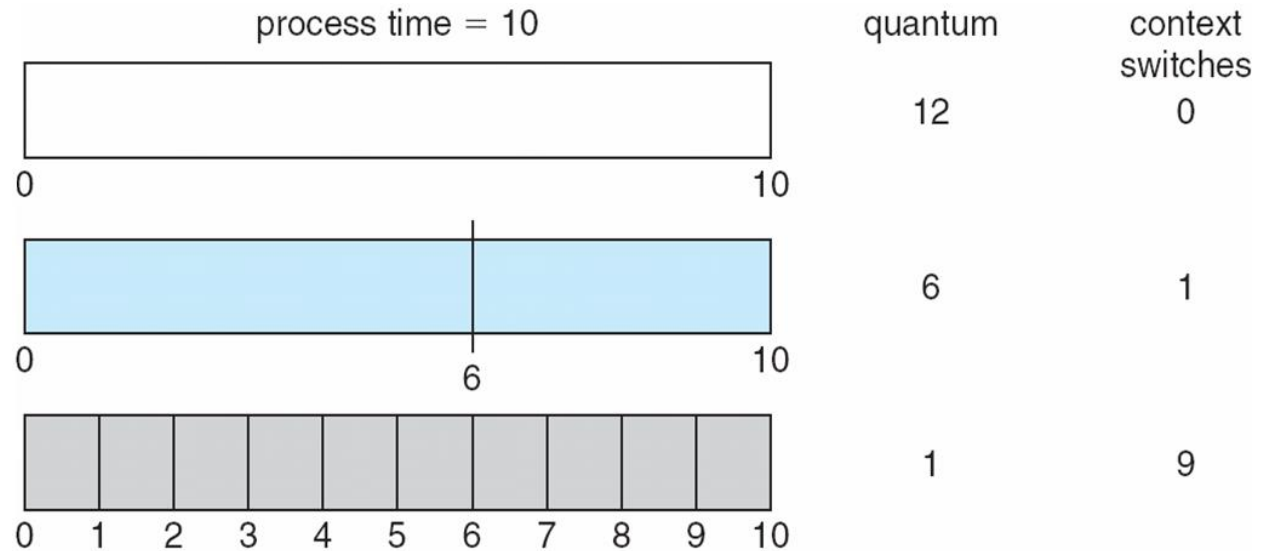
Typically, higher average turnaround than SJF, but better response



Scheduling Algorithms

Time Quantum and Context Switch Time

- The effect of context switching on the performance of RR scheduling

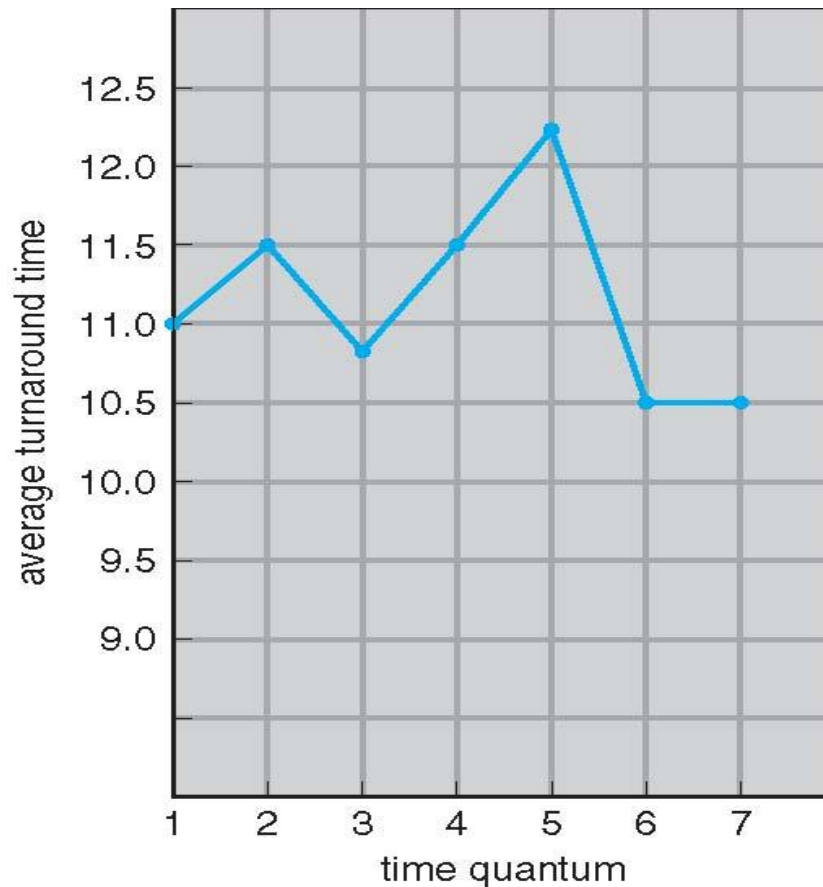


- typically the context-switch time is a small fraction of the time quantum
 - ✓ usually: time quantum: 10 ~ 100ms & context switch time: 10 μ s



Scheduling Algorithms

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7



Scheduling Algorithms

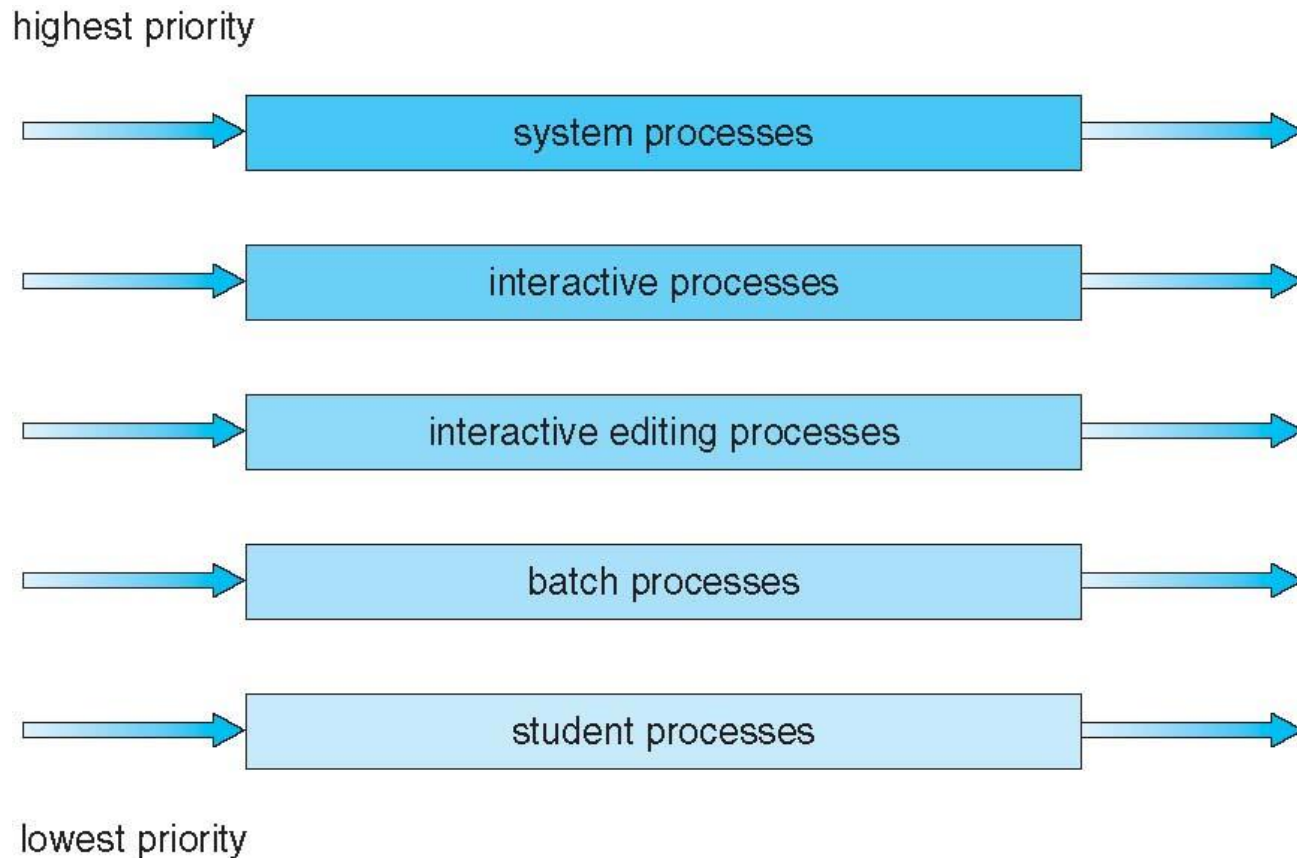
❖ Multilevel Queue (多级队列)

- ❖ Ready queue is partitioned into separate queues:
 - foreground (interactive)
 - background (batch)
- ❖ Each queue has its own scheduling algorithm
 - ✓ foreground - RR
 - ✓ background - FCFS
- ❖ Scheduling must be done between the queues
 - ✓ Fixed priority 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
 - 20% to background in FCFS



Scheduling Algorithms

❁ Multilevel Queue Scheduling





Scheduling Algorithms

✚ Multilevel Feedback Queue (多级反馈队列) Scheduling

- ✚ A process can **move** between the various queues; aging can be implemented this way
- ✚ Multilevel-feedback-queue(多级反馈队列) 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



Scheduling Algorithms

✚ Example of Multilevel Feedback Queue

✚ Three queues:

- ✓ Q0 - RR with time quantum 8 milliseconds
- ✓ Q1 - RR time quantum 16 milliseconds
- ✓ Q2 - FCFS

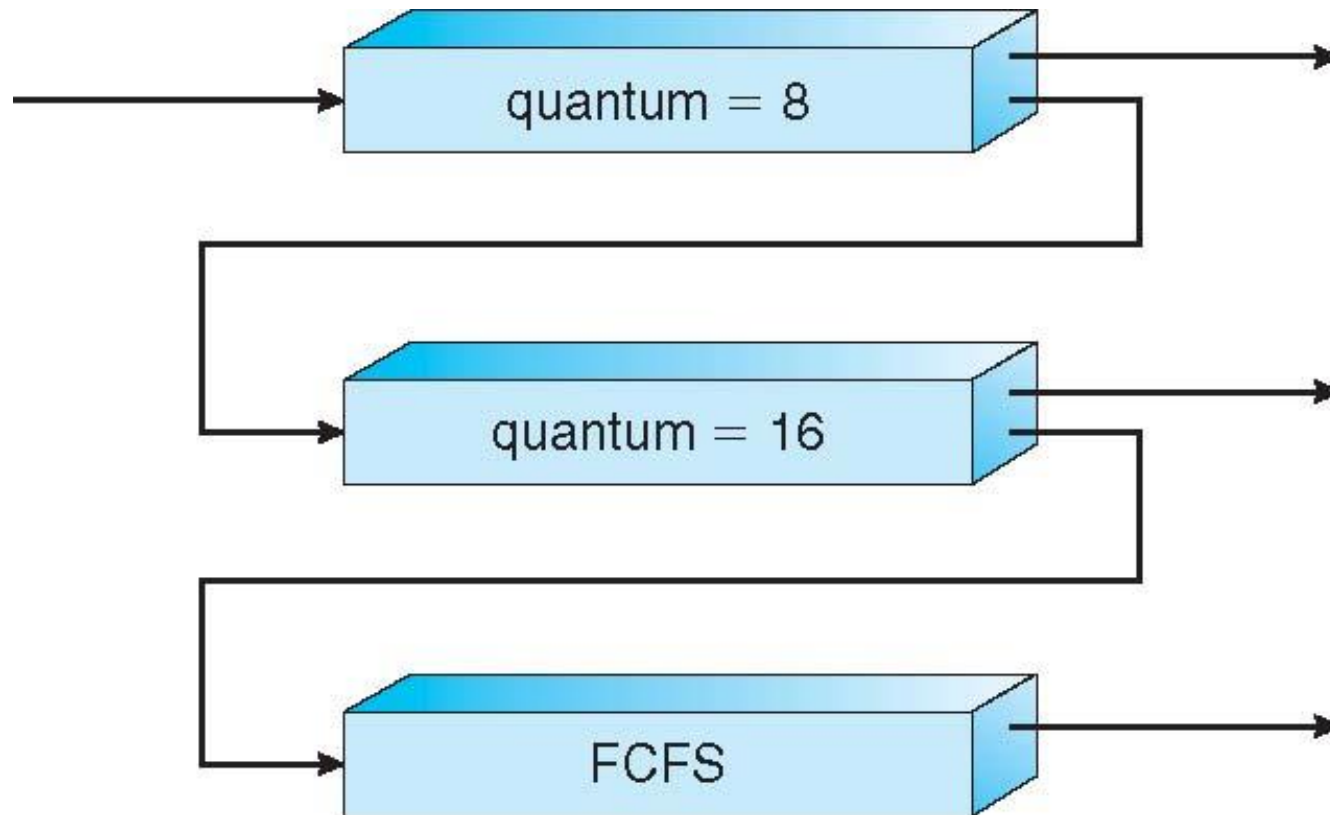
✚ Scheduling

- ✓ A new job enters queue Q0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q1.
- ✓ At Q1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q2.



Scheduling Algorithms

✿ Example of Multilevel Feedback Queue





Catalog Description

- ⊕ Basic Concepts
- ⊕ Scheduling Criteria
- ⊕ Scheduling Algorithms
- ⊕ Algorithm Evaluation



Algorithm Evaluation

Algorithm Evaluation

- ❏ How do we select a CPU scheduling algorithm for a particular system?
 - ✓ firstly, which criteria? What is the relative importance of these measures
 - ✓ then, evaluate the algorithms
 - Deterministic Modeling(确定性建模)
 - Queueing Models(排队模型)
 - Simulations(模拟)
 - Implementation Scheduling



Algorithm Evaluation

- ❖ Deterministic Modeling(确定性建模)
 - ❖ Analytic evaluation(分析评估法): One major class of evaluation methods
 - ✓ uses the given algorithm and the system workload to produce a formula or number that evaluates the performance of the algorithm for that workload.
 - ❖ Deterministic modeling(确定性建模) – takes a particular predetermined workload and defines the performance of each algorithm for that workload



Algorithm Evaluation

❖ Queuing Models (排队模型)

- ❖ Usually, two distributions can be measured and then approximated or simply estimated
 - ✓ the distribution of CPU and I/O bursts
 - ✓ the arrival-time distribution
- ❖ Queueing-network analysis (排队网络分析)
 - ✓ Computer System: a network of servers, each server has a queue of waiting processes
 - CPU: ready queue;
 - I/O: device queues (\equiv waiting queue)
 - ✓ Given arriving rates and service rates
 \Rightarrow utilization,
average queue length, average wait time, ...



Algorithm Evaluation

❖ Simulations (模拟)

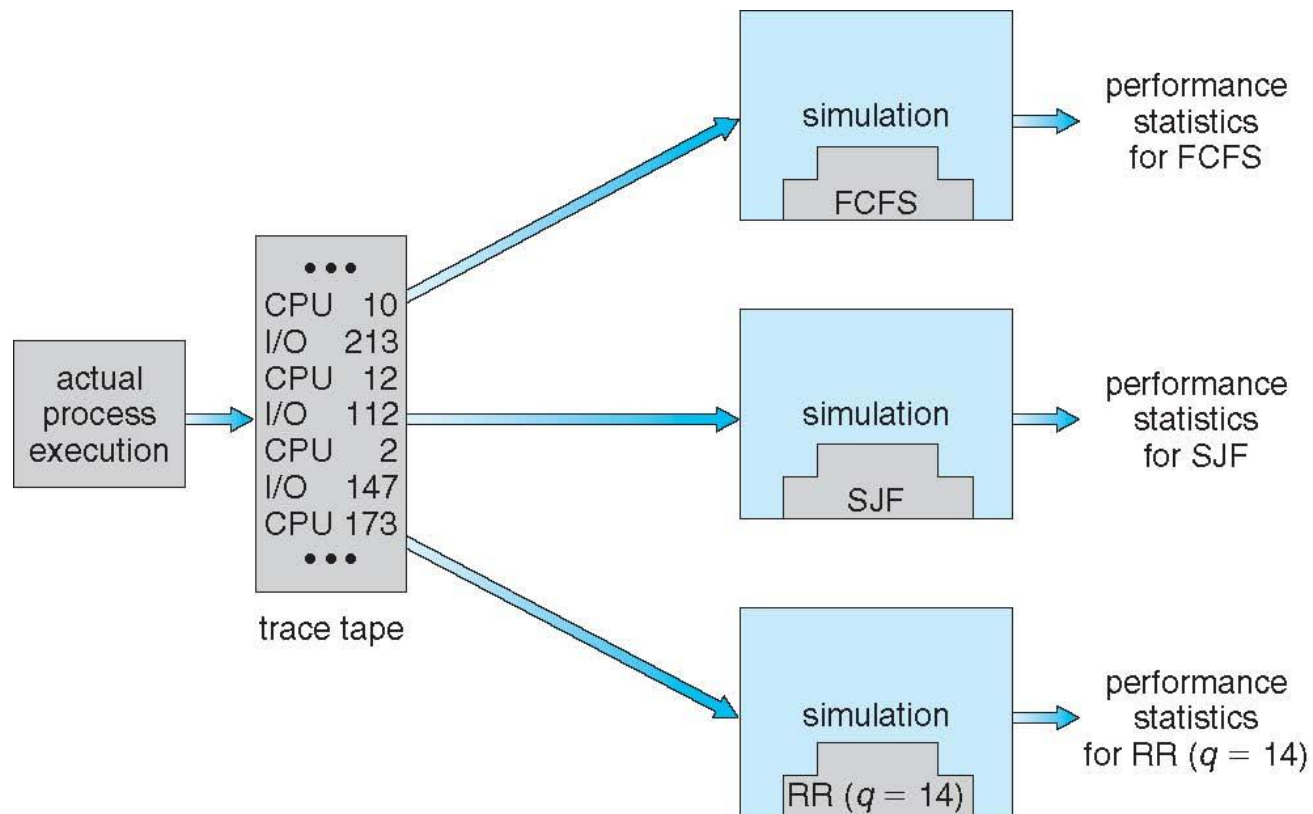
- ❖ Running simulations involves programming a model of the computer system.
 - ✓ Software data structures represent the major components
 - a clock
 - the system state is modified to reflect the activities of the devices, the processes and the scheduler.
 - ✓ finally, the statistics are gathered
- ❖ How to generate the data to drive the simulation?
 - ✓ distribution-driven simulation random-number generator, according to probability distributions, to generate processes, CPU burst times, arrivals, departures, ...
 - The distributions can be defined mathematically (uniform, exponential, Poisson) or empirically
 - may be inaccurate



Algorithm Evaluation

❁ Evaluation of CPU schedulers by Simulation

❁ trace tapes (跟踪磁带)





Algorithm Evaluation

❖ Implementation

- ❖ This approach put the actual algorithm in the real system for evaluation under real operating conditions
- ❖ the main difficulty: high cost