Operating System Concepts

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information Engineering, Chang Gung University

Contents

1. Introduction
2. System Structures
3. Process Concept
4. Multithreaded Programming
5. Process Scheduling
6. Synchronization
7. Deadlocks
8. Memory-Management Strategies
9. Virtual-Memory Management
10. File System
11. Implementing File Systems
12. Secondary-Storage Systems

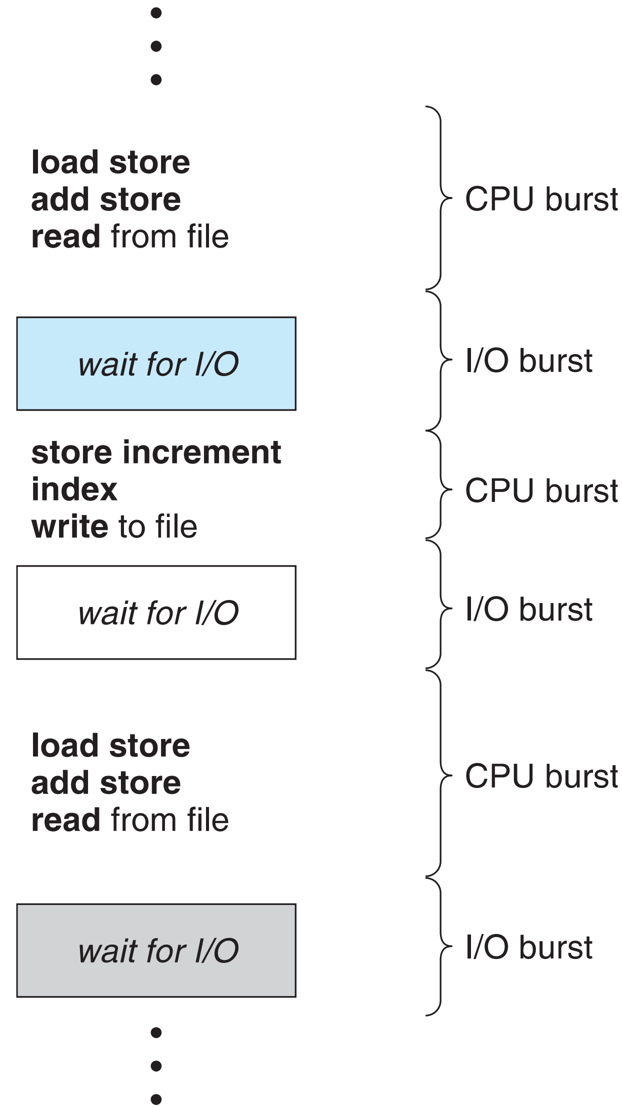
Chapter 5. Process Scheduling

Objectives

- To introduce CPU scheduling, which is the basis for multi-programmed 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

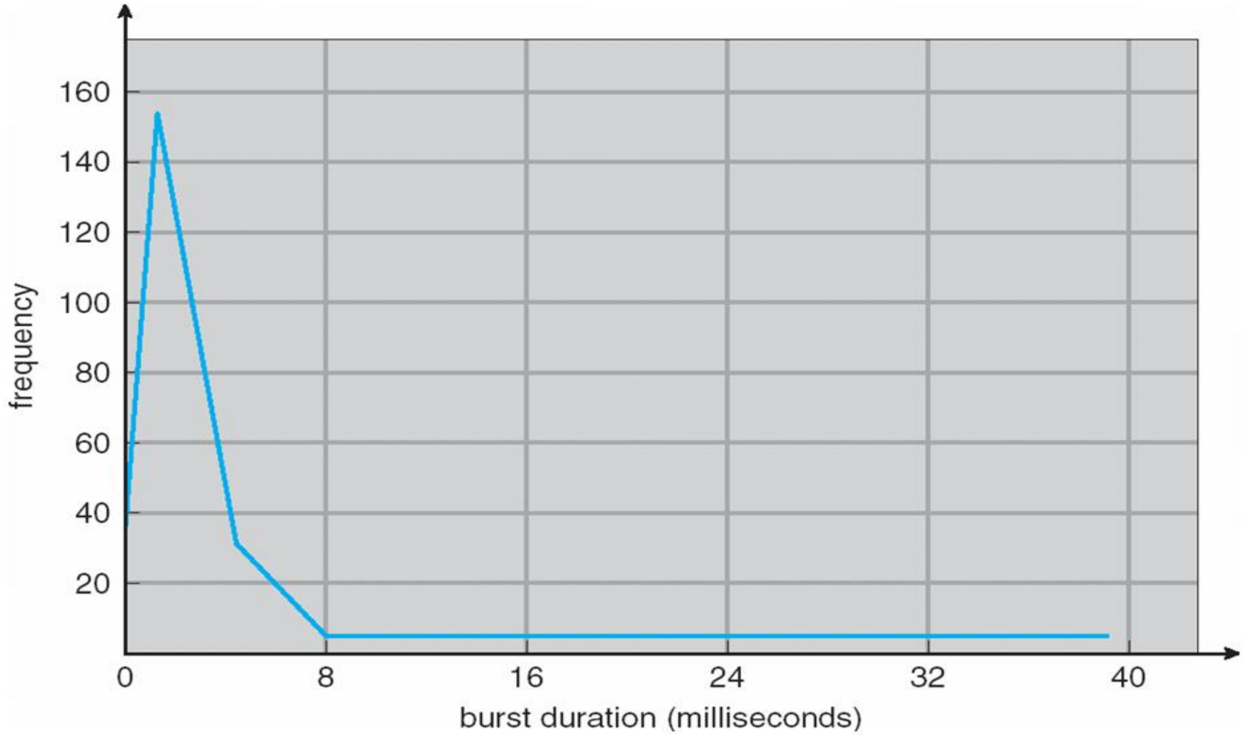
Basic Concepts

- CPU–I/O Burst Cycle
  - Process execution consists of a cycle of CPU execution and I/O waiting
- Process Execution
  - CPU-bound programs tend to have a few very long CPU bursts
  - IO-bound programs tend to have many very short CPU bursts



Histogram of CPU-burst Times

- The distribution can help in selecting an appropriate CPU scheduling algorithms



CPU Scheduler

- Short-term scheduler selects a process among the processes in the ready queue, and allocates the CPU to the selected process
  - 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

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to resume that process
- Dispatch latency – the time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

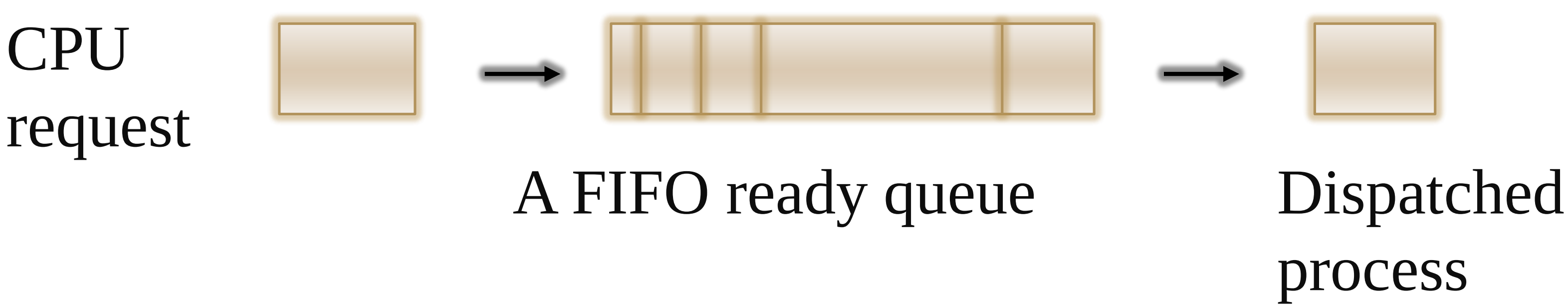
- Why?
  - Different scheduling algorithms may favor one class of processes over another
- Criteria
  - CPU Utilization
  - Throughput
  - Turnaround Time: (Completion Time) – (Start Time)
  - Waiting Time: Waiting in the Ready Queue
  - Response Time: First Response Time

Scheduling Algorithms

- First-Come, First-Served Scheduling (FIFO)
- Shortest-Job-First Scheduling (SJF)
- Priority Scheduling
- Round-Robin Scheduling (RR)
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling
- Multiple-Processor Scheduling

First-Come, First-Served (FCFS) Scheduling

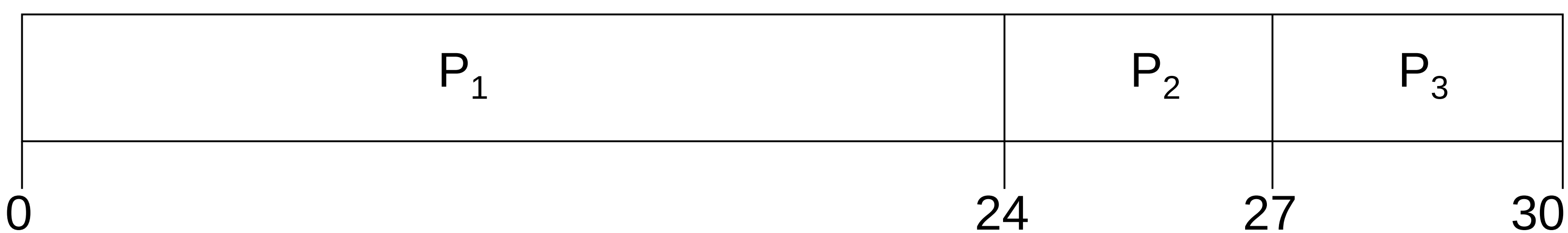
- The process which requests the CPU first is allocated the CPU
- Properties:
  - Non-preemptive scheduling
  - CPU might be hold for an extended period



A Scheduling Example of FCFS (1/2)

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



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

A Scheduling Example of FCFS (2/2)

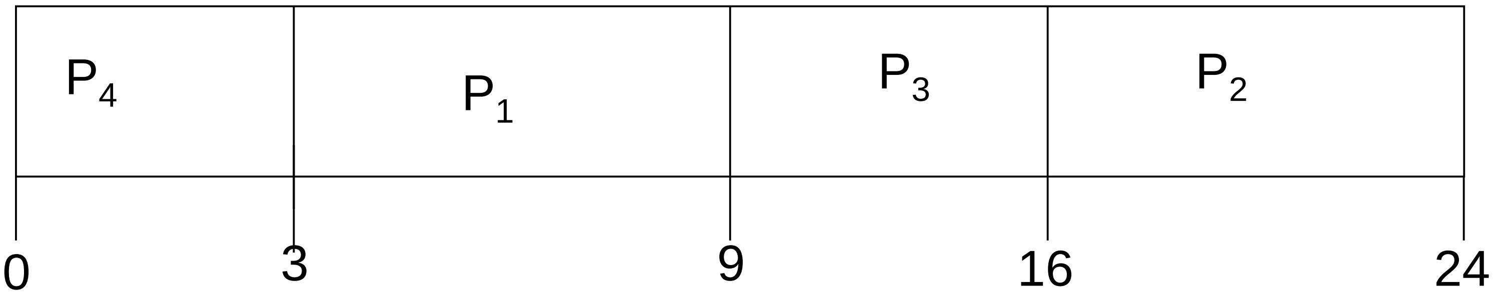
- Suppose that the processes arrive in the order:
  - P2 , P3 , P1
- The Gantt chart for the schedule is:

- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- **Convoy effect** – short processes behind long a process

Shortest-Job-First (SJF) Scheduling

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

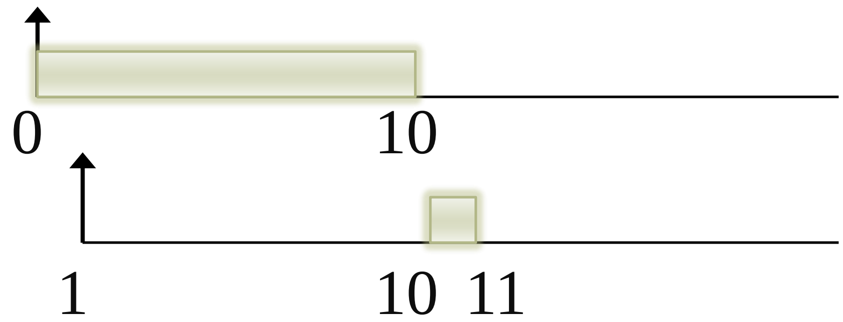
SJF Scheduling Analysis

- Non-preemptive SJF scheduling is optimal when processes are all ready at time 0
  - The minimum average waiting time
- It is difficult to know the length of the next CPU request
  - Prediction of the next CPU burst time using exponential averaging

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

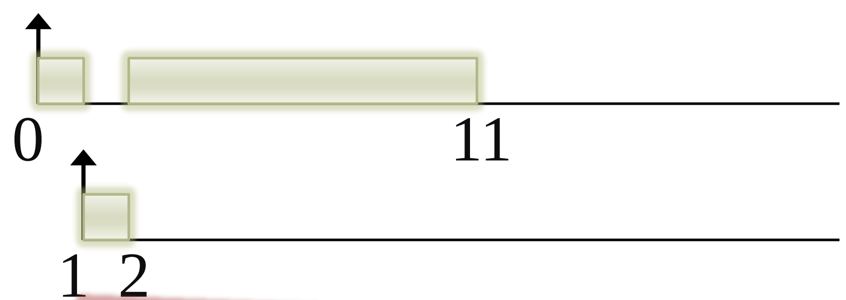
Preemptive SJF Scheduling

- Preemptive or Non-preemptive?
  - Criteria such as AWT (Average Waiting Time)



Non-preemptive
AWT = (0+(10-1))/2
= 9/2 = 4.5

or



Preemptive AWT
= ((2-1)+0) = 0.5

Shortest-Remaining-Time-First Scheduling

Department of Computer Science and Information Engineering, Chang Gung University

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority
- Priority Assignment
  - Internally defined – use some measurable quantity, such as the number of open files, $\frac{\text{Average CPU Burst}}{\text{Average I/O Burst}}$
  - Externally defined – set by criteria external to the OS, such as the criticality levels of jobs

A Scheduling Example with Priority Scheduling

Process	CPU Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Gantt Graph

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

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

Issues of Priority Scheduling

- Problem: Starvation – low priority processes may never execute
- Solution: Aging – as time progresses increase the priority of the process
- A Special Case: SJF is priority scheduling where priority is the inverse of predicted next CPU burst time

Round Robin (RR) Scheduling

- 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
  - 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

A Scheduling Example of RR Scheduling

Process	CPU Burst Time
P1	24
P2	3
P3	3

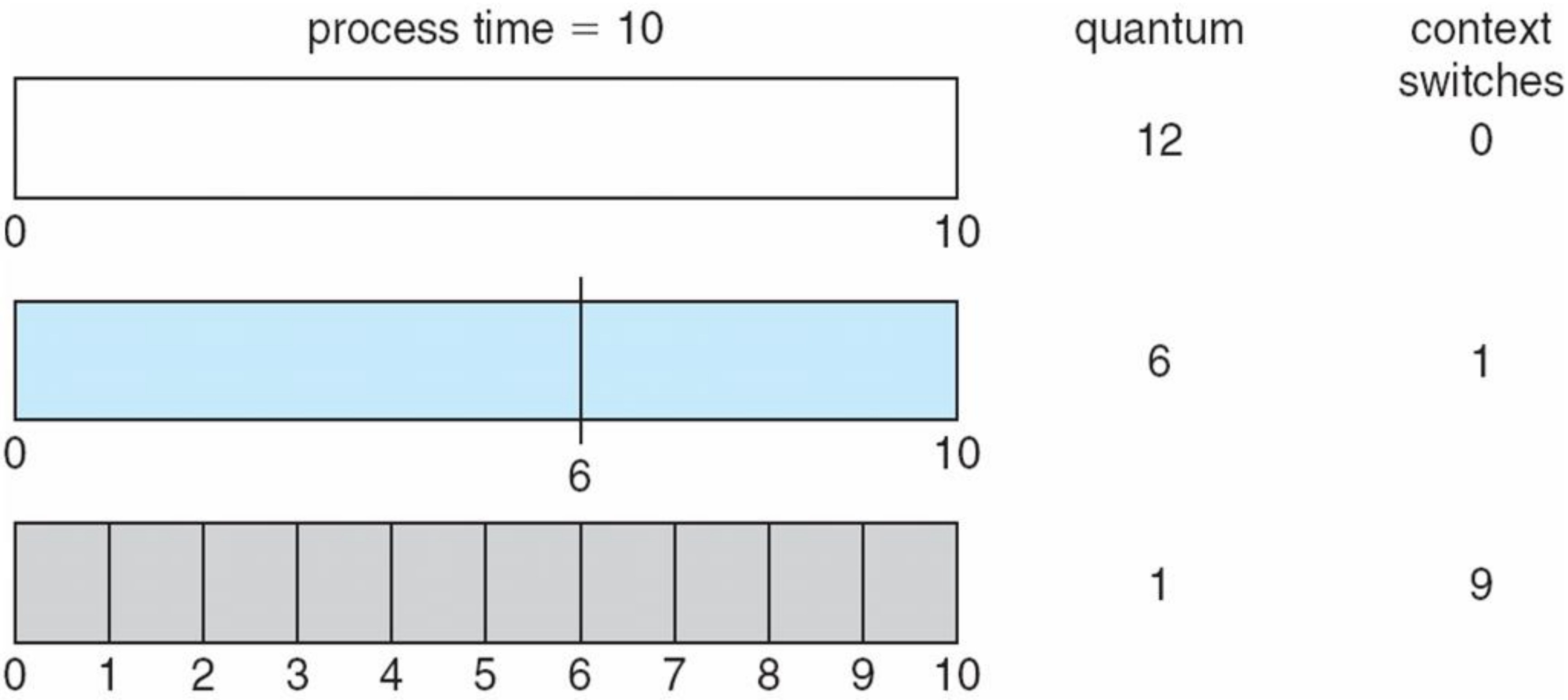
Time slice = 4

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

0 4 7 10 14 18 22 26 30

AWT = ((10-4) + (4-0) + (7-0))/3 = 17/3 = 5.66

Time Quantum and Context Switch



Issues of RR Scheduling

- Time quantum too large $\rightarrow$ FIFO
- Time quantum too small $\rightarrow$ Time quantum must be large with respect to context switch time, otherwise overhead is too high
  - Time quantum usually 10 ms to 100ms
  - Context switch < 10 μs
- A rule of thumb is that 80 percent of the CPU bursts should be shorter than the time quantum

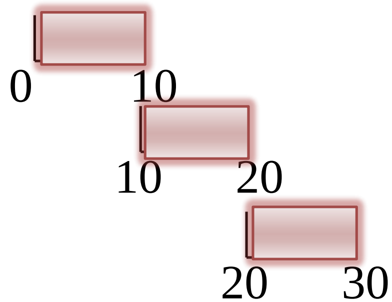
Issues of RR Scheduling — Turnaround Time

process (10ms) quantum = 10 quantum = 1

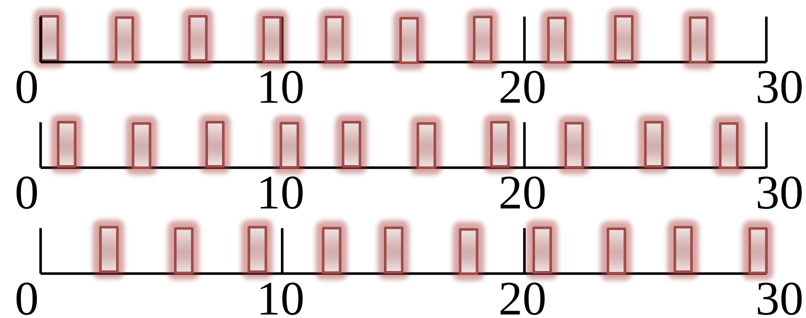
P1

P2

P3



Average Turnaround Time
= (10+20+30)/3 = 20

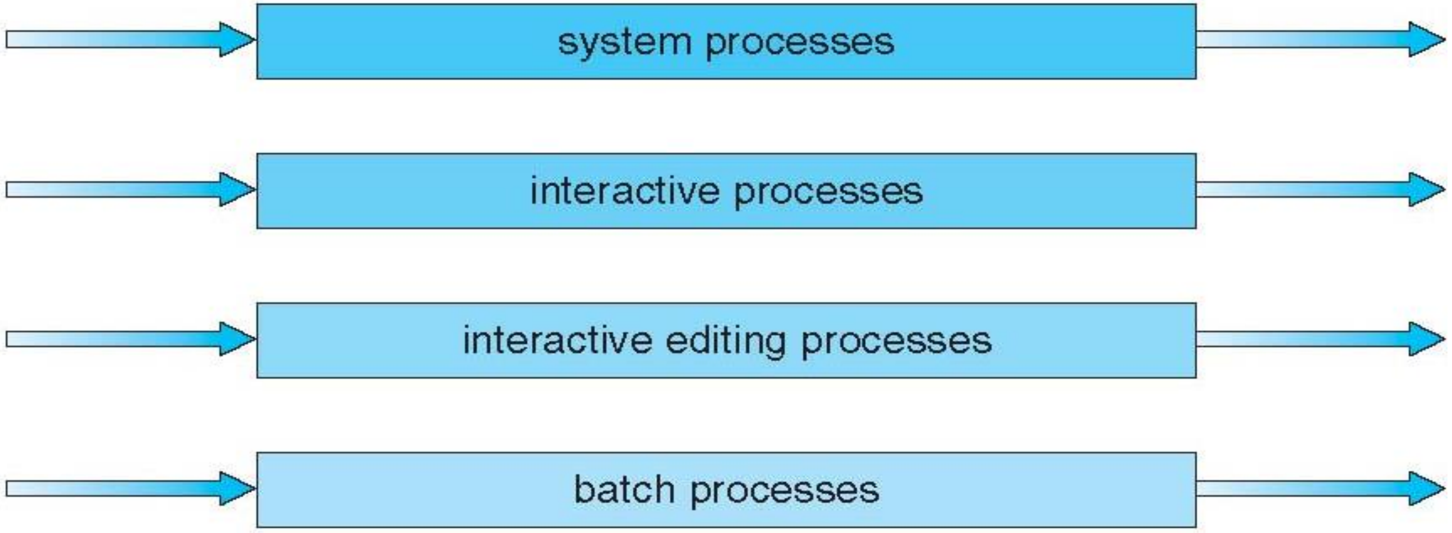


ATT = (28+29+30)/3 = 29

=> 80% CPU Burst < time slice

Multilevel Queue Scheduling

- Partition the ready queue into several separate queues
  - ➔ Processes can be classified into different groups and permanently assigned to one queue



Multilevel Queue Scheduling

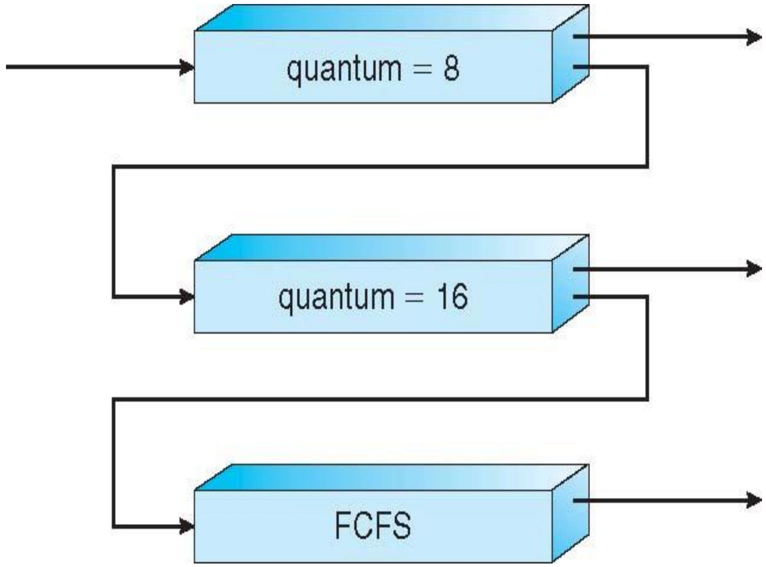
- Intra-queue scheduling
  - Independent choice of scheduling algorithms
    - e. g., foreground – RR, and background – FCFS
- Inter-queue scheduling
  - Fixed-priority preemptive scheduling
    - e.g., foreground queues always have absolute priority over the background queues
  - Time slice between queues
    - e.g., 80% CPU is given to foreground processes, and 20% CPU to background processes

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
  - The method to determine which queue a newly ready process will enter

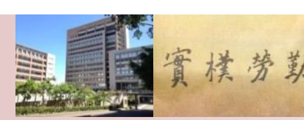
An Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - Do jobs in Q_0 first and then Q_1 and then Q_2
  - A new job enters queue Q_0
    - When it gains CPU, job receives 8 milliseconds
    - If it does not finish in 8 milliseconds, job is moved to queue Q_1
  - At Q_1 each job receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2



Thread Scheduling

- To run on a CPU, user threads must be mapped to an associated kernel thread
- Local Scheduling
  - Contention Scope: Process-Contention Scope (PCS)
  - How the threads library decides which thread to put onto an available kernel thread
- Global Scheduling
  - Contention Scope: System-Contention Scope (SCS)
  - How the kernel decides which kernel thread to run on CPU next



Multiple-Processor Scheduling

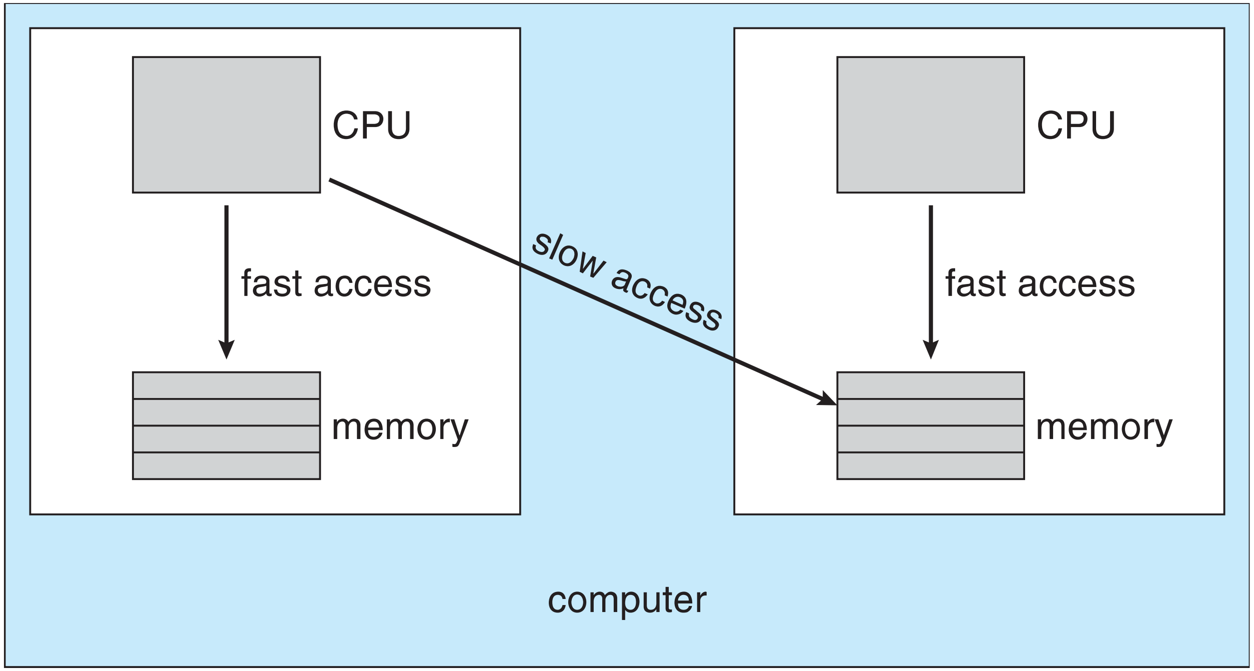
- CPU scheduling in a system with multiple CPUs
- A Homogeneous System
  - Processors are identical in terms of their functionality
- A Heterogeneous System
  - Programs must be compiled for instructions on proper processors

Homogeneous Processors

- **Asymmetric multiprocessing** – only one processor accesses the system data structures, alleviating the need for data sharing
- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each processor has its own private queue of ready processes

Multiple-Processor Scheduling—Processor Affinity

- A process might prefer to run on specific processors
  - Hard affinity: `sched_setaffinity()`
  - Soft affinity: non-uniform memory access



Multiple-Processor Scheduling—Load Balancing

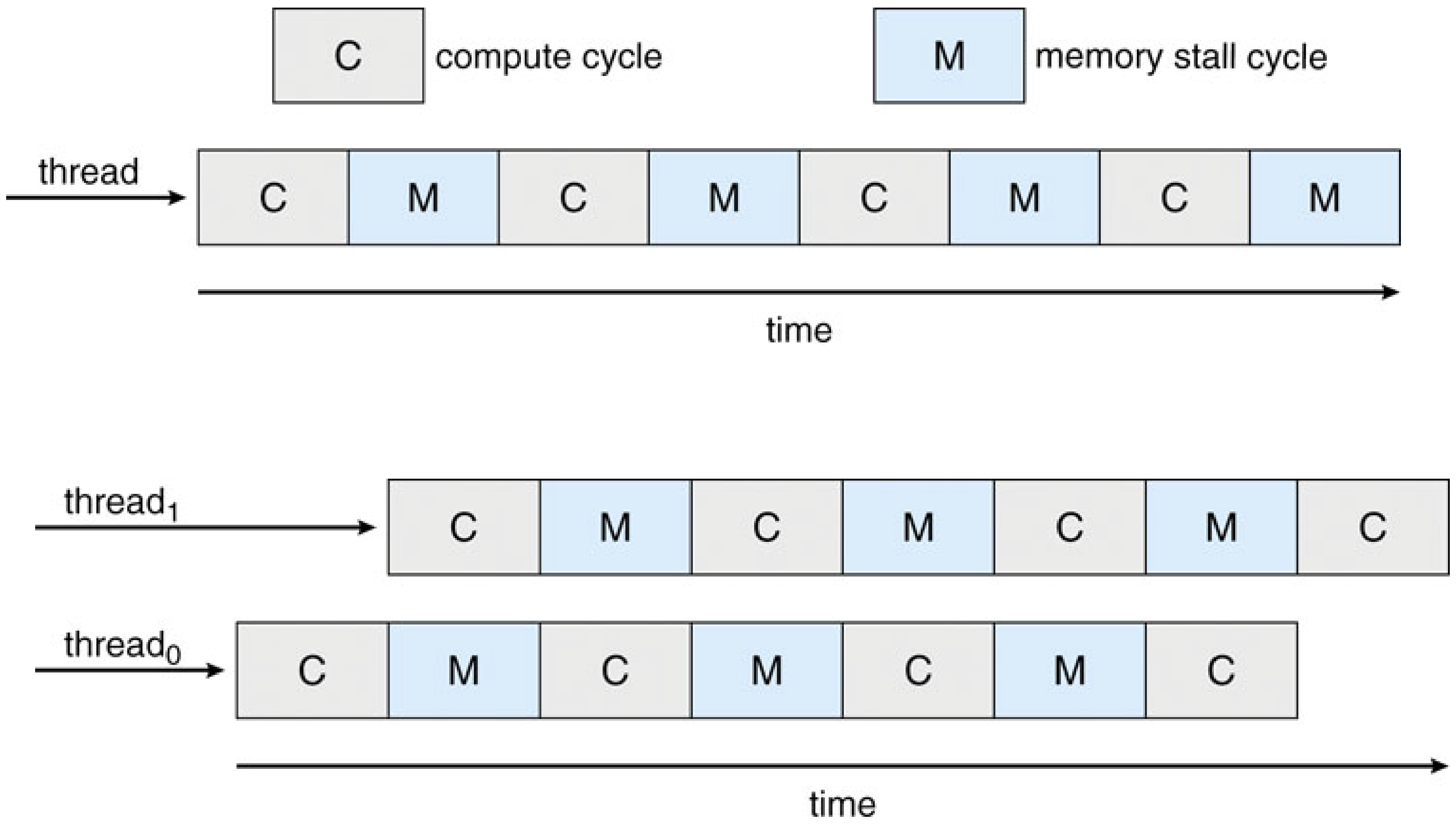
- Attempt to keep the workload evenly distributed across all processors in an SMP system
- Push migration
  - A specific task periodically checks the load on each processor and evenly distributes the load by moving processes from overloaded to idle or less-busy processors
- Pull migration
  - An idle processor pulls a waiting task from a busy processor

Multicore Processors

- Multicore Processor: A physical chip with multiple processor cores.
- Scheduling Issues:
  - Memory Stall
    - Coarse-Grained Multithreading
      - Thread execution until a long latency
    - Fine-Grained Multithreading
      - Better architecture design for switching

→ Multiple Hardware Threads

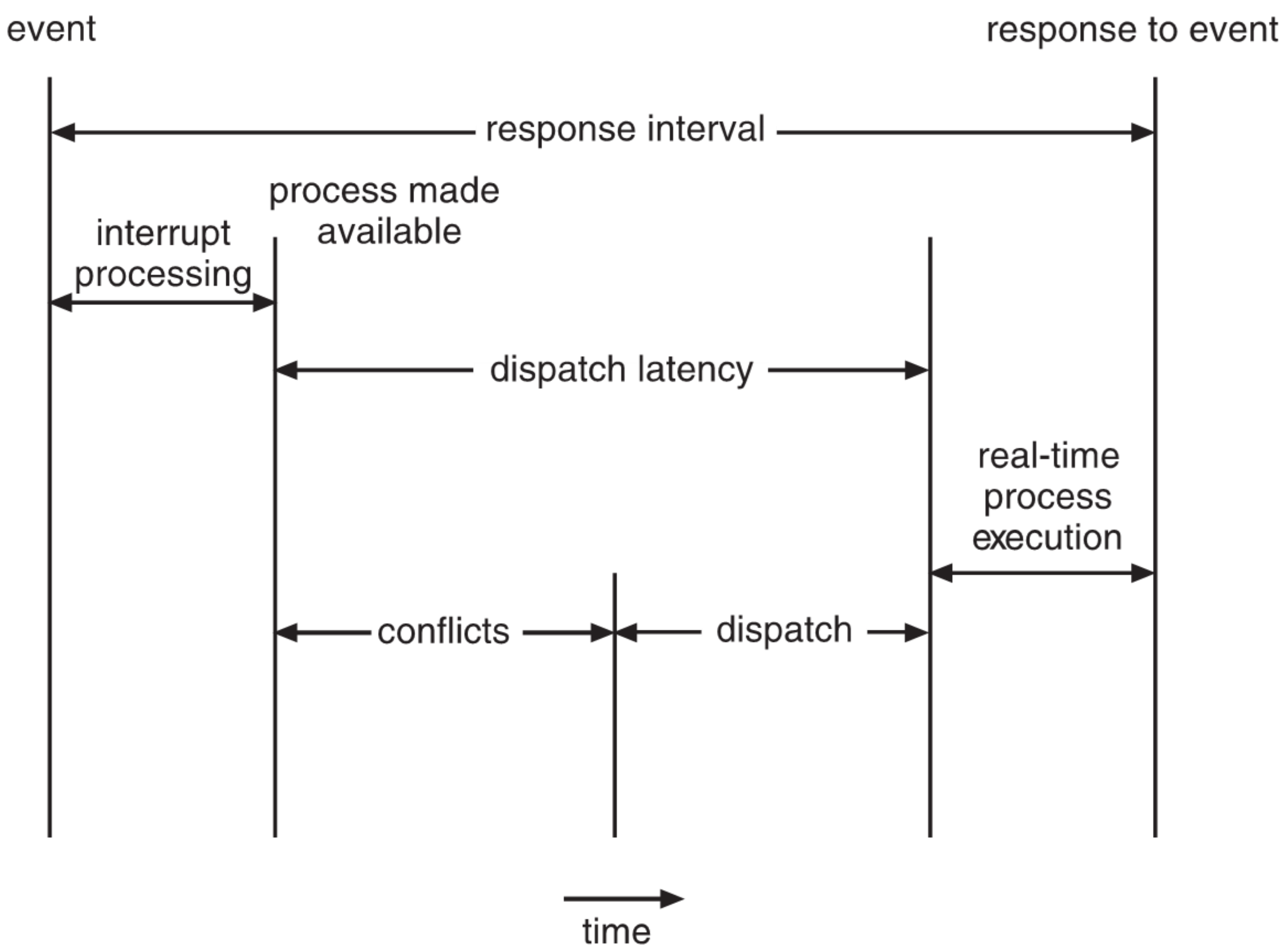
Multithreaded Multicore (Hyper-Threading) System



Real-Time Scheduling (1/2)

- Each task (process) has to be completed before its **deadline**
- **Soft real-time systems** – try to serve a real-time task by its deadline
- **Hard real-time systems** – a real-time task must be served by its deadline
- Two types of latencies affect performance
  1. Interrupt latency – time from arrival of interrupt to start of routine that serves the interrupt
  2. Dispatch latency – time for schedule to take current process off CPU and switch to another

Real-Time Scheduling (2/2)



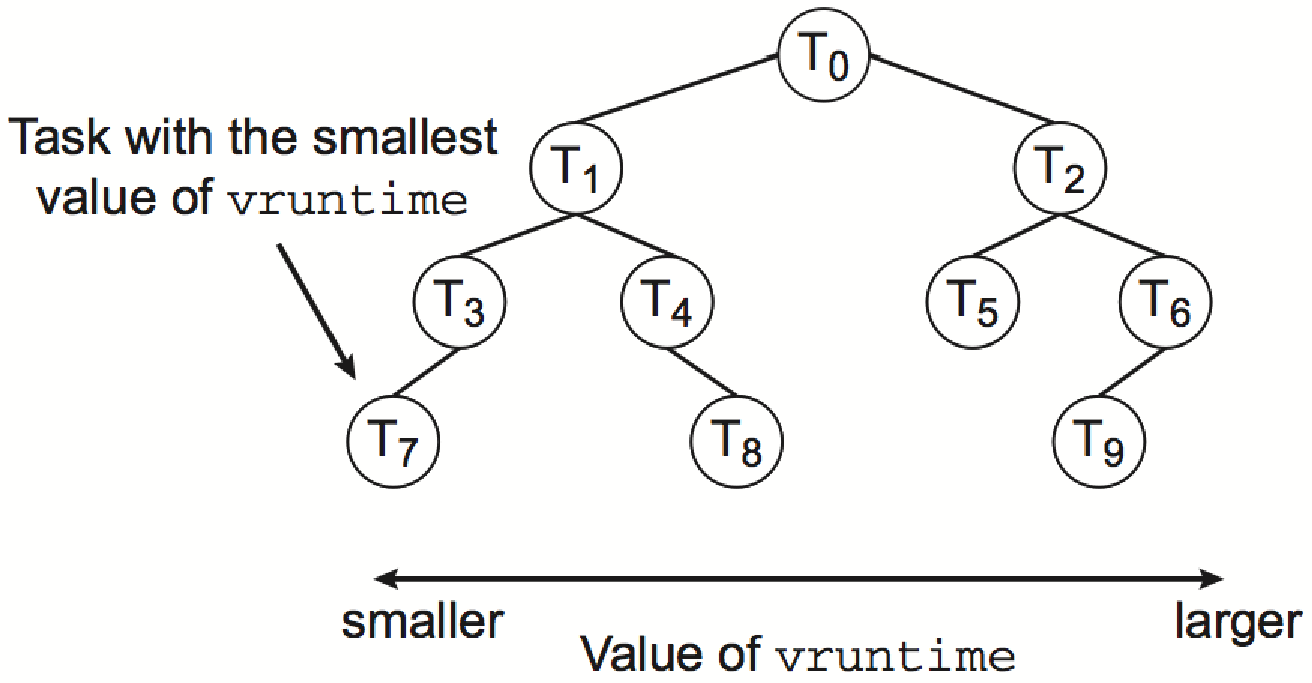
Operating System Examples – Linux in Version 2.6.23 + (1/3)

- Completely Fair Scheduler (CFS)
  - CFS scheduler maintains per task **virtual run time** in variable **`vruntime`**
  - Associated with decay factor based on priority of task:
    - lower priority ➔ higher decay rate
  - Normal default priority yields virtual run time = actual run time
  - To decide next task to run, scheduler picks task with lowest virtual run time
- Nice Value
  - From -20 to +19
  - Lower value is higher priority

實 樸 勞 勤

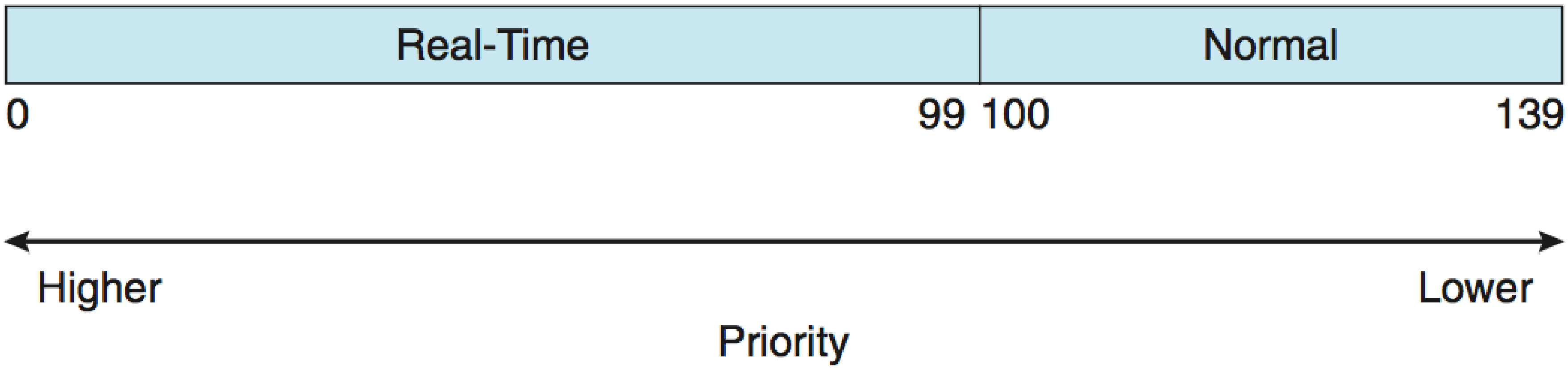
Operating System Examples – Linux in Version 2.6.23 + (2/3)

- A red-back tree is used to maintain the virtual run times of tasks



Operating System Examples – Linux in Version 2.6.23 + (3/3)

- Real-time scheduling according to POSIX
  - Real-time tasks have static priorities
- Real-time plus normal map into global priority scheme
- Nice value of -20 maps to global priority 100
- Nice value of +19 maps to priority 139



Operating System Examples – Windows Scheduling (1/3)

A Typical Class

Base Priority

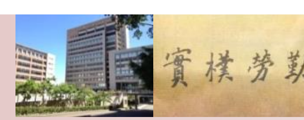
	Real-time	High	Above normal	Normal	Below normal	Idle priority
Time-critical	31	15	15	15	15	15
Highest	26	15	12	10	8	6
Above normal	25	14	11	9	7	5
Normal	24	13	10	8	6	4
Below normal	23	12	9	7	5	3
Lowest	22	11	8	6	4	2
Idle	16	1	1	1	1	1

Real-Time Class

Variable Class (1..15)

Operating System Examples – Windows Scheduling (2/3)

- Priority-Based Preemptive Scheduling
  - Priority Range: from 0 to 31
    - Variable class uses 1-15
    - Real-time class uses 16-31
  - Dispatcher: A process runs until
    - It is preempted by a higher-priority process
    - It terminates
    - Its time quantum ends
    - It calls a blocking system call
  - Idle thread
- A Queue per Priority Level



Operating System Examples – Windows Scheduling (3/3)

- Each thread has a base priority that represents a value in the priority range of its class
- Priority Changing
  - Increased after some waiting
    - Different amount for different I/O devices
  - Decreased after some computation
    - The priority is never lowered below the base priority
- Favor foreground processes
  - Each foreground task is given more time quantum (typically 3 times longer)

Scheduling Algorithm Evaluation

- A General Procedure
  - Select criteria that may include several measures, e.g., maximize CPU utilization while confining the maximum response time to 1 second
  - Evaluate various algorithms
- Evaluation Methods:
  - Deterministic modeling
  - Queuing models
  - Simulation
  - Implementation

Deterministic Modeling

- A Typical Type of Analytic Evaluation
  - Take a particular predetermined workload and defines the performance of each algorithm for that workload
- Properties
  - Simple and fast
  - Through excessive executions of a number of examples, trends might be identified
  - But it needs exact numbers for inputs, and its answers only apply to those cases
    - Being too specific and requires too exact knowledge to be useful

Deterministic Modeling

process	CPU Burst time
P1	10
P2	29
P3	3
P4	7
P5	12

FCFS

P1	P2	P3	P4	P5

0 10 39 42 49 61

Average Waiting Time (AWT)=(0+10+39+42+49)/5=28

Nonpreemptive Shortest Job First

P3	P4	P1	P5	P2

0 3 10 20 32 61

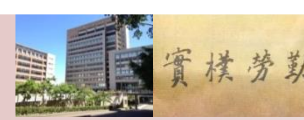
AWT=(10+32+0+3+20)/5=13

Round Robin (quantum =10)

P1	P2	P3	P4	P5	P2	P5	P2

0 10 2023 30 40 5052 61

AWT=(0+(10+20+2)+20+23+(30+10))/5=23



Queuing Models

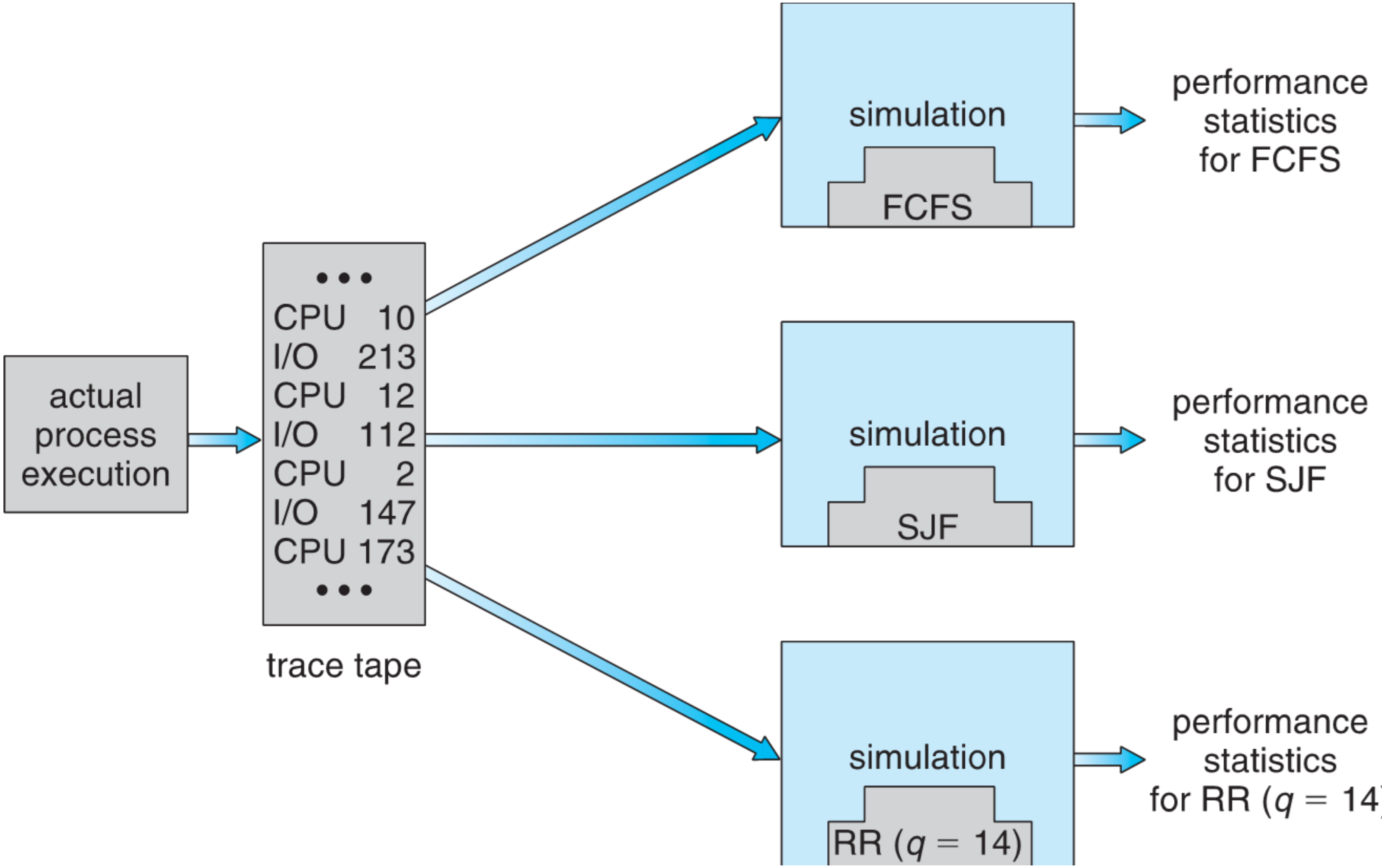
- Motivation:
  - Workloads vary, and there is no static set of processes
- Models (~ Queuing-Network Analysis)
  - Workload:
    - Arrival rate: the distribution of times when processes arrive
    - The distributions of CPU & I/O bursts
  - Service rate

Simulation (1/2)

- Motivation:
  - Get a more accurate evaluation
- Procedures:
  - Program a model of the computer system
  - Drive the simulation with various data sets
    - Randomly generated according to some probability distributions

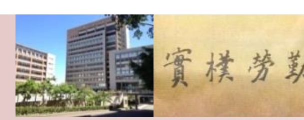
      ➔ Inaccuracy occurs because of only the occurrence frequency of events. Miss the order & the relationships of events.
    - Trace tapes: monitor the real system & record the sequence of actual events.

Simulation (2/2)



Implementation

- Motivation:
  - Get more accurate results than a simulation
- Procedure:
  - Code scheduling algorithms
  - Put them in the OS
  - Evaluate the real behaviors
- Difficulties:
  - Cost in coding algorithms and modifying the OS
  - Reaction of users to a constantly changing the OS
  - The environment in which algorithms are used will change



Quiz

(40%) 考慮在時間點0的時候已經就緒的五個工作，到達的順序為P1, P2, P3, P4, P5。使用兩個排程演算法FCFS (First-Come, First-Served)以及SJF (Shortest-Job-First) 來排程。(1)請畫下兩個排程演算法的排程圖，(2)請分別算出兩個排程演算法中每個工作的等待時間，若無算式一率不給分(算式可以只是簡單的加減法運算)。

Process	Burst Time
P_1	10 ms
P_2	1 ms
P_3	2 ms
P_4	6 ms
P_5	3 ms

- 其他考題：
  - FIFO, Preemptive and Nonpreemptive SJF, RR
  - Turnaround Time, Waiting Time, Response Time: