Operating Systems
[1. Introduction]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

CSIE Department

National Taiwan University

From Operating System Concepts (10th Edition)

Objectives

- ❑ Describe the general organization of a computer system and the role of interrupts
- ❑ Describe the components in a modern multiprocessor computer system
- ❑ Illustrate the transition from user mode to kernel mode
- ❑ Discuss how operating systems are used in various computing environments

Outline

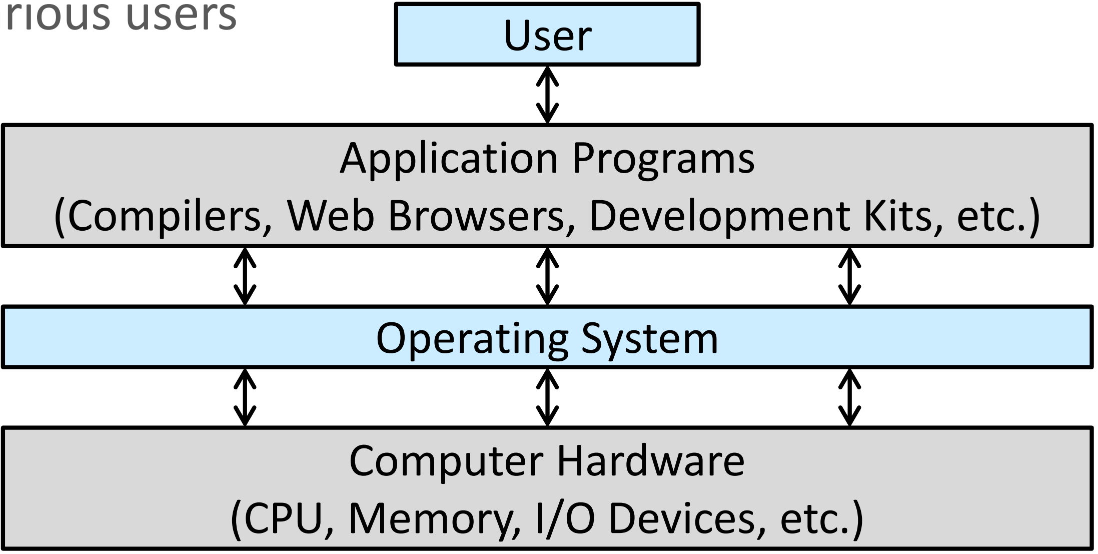
- ❑ **What Operating Systems Do**
  - ➢ User View, System View, Defining Operating Systems
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

What is an Operating System?

- An **operating system** is software
  - Manage a computer's hardware
  - Provide a basis for application program
  - Act as an intermediary between the computer user and the computer hardware
- Operating systems are everywhere
  - Cars
  - Home appliances
  - Smart phones
  - Personal computers
  - Enterprise computers
  - Cloud computing environments

Components of Computer Systems

- ❑ Hardware
  - ➢ Provide the basic computing resources for the system
- ❑ Application programs
  - ➢ Define the ways in which these resources are used to solve users computing problems
- ❑ Operating system
  - ➢ Control and coordinate the hardware among the various application programs for the various users
- ❑ User


User
Application Programs
(Compilers, Web Browsers, Development Kits, etc.)
Operating System
Computer Hardware
(CPU, Memory, I/O Devices, etc.)

Outline

- **What Operating Systems Do**
  - **User View**, System View, and Defining Operating Systems
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management, Security and Protection, Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- Free and Open-Source Operating Systems

User View

- Many computer users sit with a laptop or in front of a PC
  - The operating system is designed mostly for **ease of use**
  - Some attention is paid to performance and security
  - None (or not too much) is paid to **resource utilization**: how various hardware and software resources are shared
- Increasing users interact with mobile devices
  - Connected to networks through cellular or other wireless technologies
  - Featured a touch screen and/or a voice recognition interface
- Some computers have little or no user view
  - They are designed primarily to run without user intervention
  - Examples: embedded computers in home devices and automobiles

Outline

- ❑ **What Operating Systems Do**
  - ➢ User View, **System View**, Defining Operating Systems
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

System View

- ❑ A **resource allocator**
  - ➢ The operating system is the program most intimately involved with the hardware
  - ➢ The operating system must allocate resources to specific programs and users to make the computer system efficiently and fairly
    - Requests for resources are numerous and possibly conflicting
- ❑ A **control program**
  - ➢ A slightly different view emphasizes the need to control the various I/O devices and user programs
  - ➢ The operating system manages the execution of user programs to prevent errors and improper use of the computer

Outline

- ❑ **What Operating Systems Do**
  - ➢ User View, System View, **Defining Operating Systems**
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Defining Operating Systems (1/3)

- History of computers
- No completely adequate definition of an operating system
  - Operating systems exist because they offer a reasonable way to solve the problem of creating a usable computing system
    - Bare hardware alone is not particularly easy to use
    - Application programs require certain common operations
- No universally accepted definition of what is part of the operating system

Defining Operating Systems (2/3)

- A more common definition
- The one program running at all times on the computer, usually called the **kernel**
- Two other types of programs along with the kernel
  - System programs
    - Be associated with the operating system but are not necessarily part of the kernel
  - Application programs
    - Include all programs not associated with the operation of the system

Defining Operating Systems (3/3)

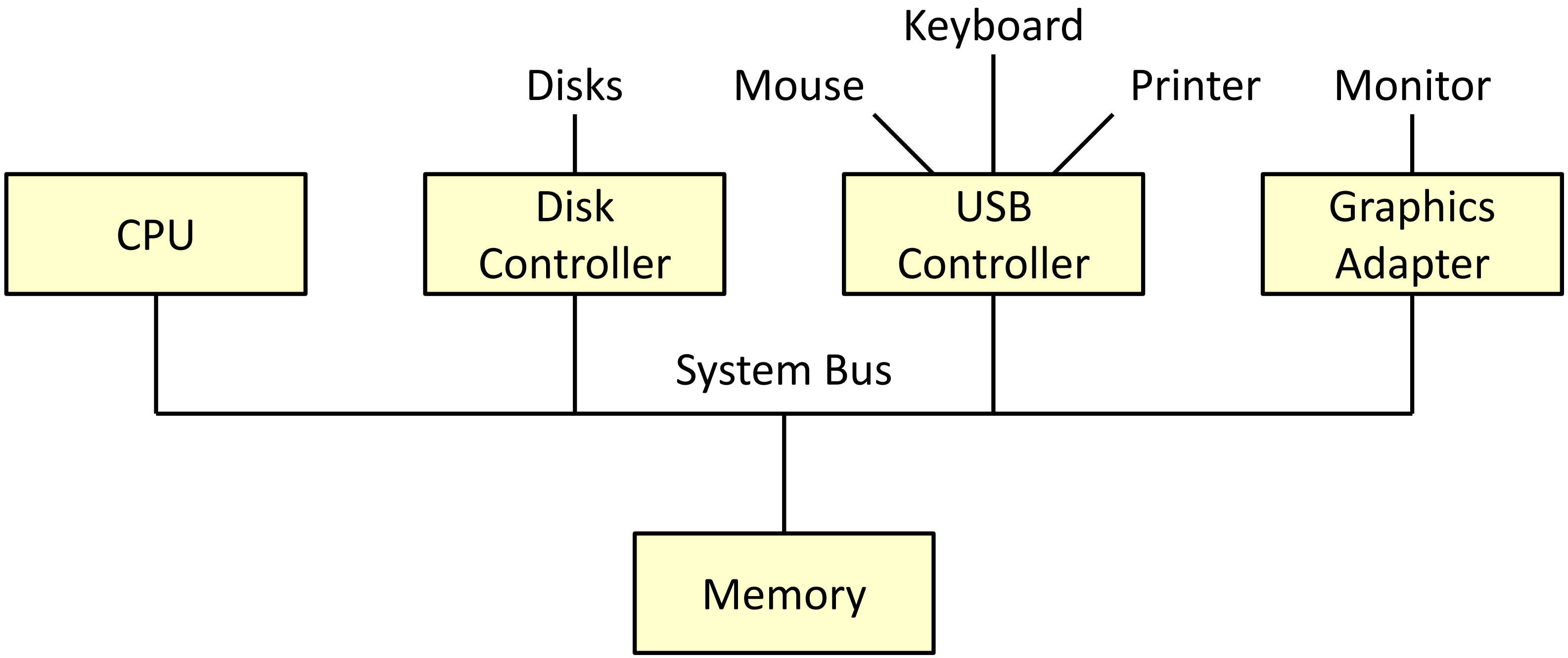
- Today, mobile operating systems often include not only a core kernel but also **middleware**
  - A set of software frameworks that provide additional services to application developers
  - Examples: databases, multimedia, graphics

Outline

- ❑ What Operating Systems Do
- ❑ **Computer-System Organization**
  - ➢ Interrupts, Storage Structure, I/O Structure
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Computer-System Organization

- A **bus** connects one or more CPUs and device controllers and provides access between components and shared memory
  - Operating systems have a device driver for each device controller
  - CPUs and device controllers can execute in parallel
  - To ensure orderly access to the shared memory, a memory controller synchronizes access to the memory



Outline

- ❑ What Operating Systems Do
- ❑ **Computer-System Organization**
  - ➢ **Interrupts**, Storage Structure, I/O Structure
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

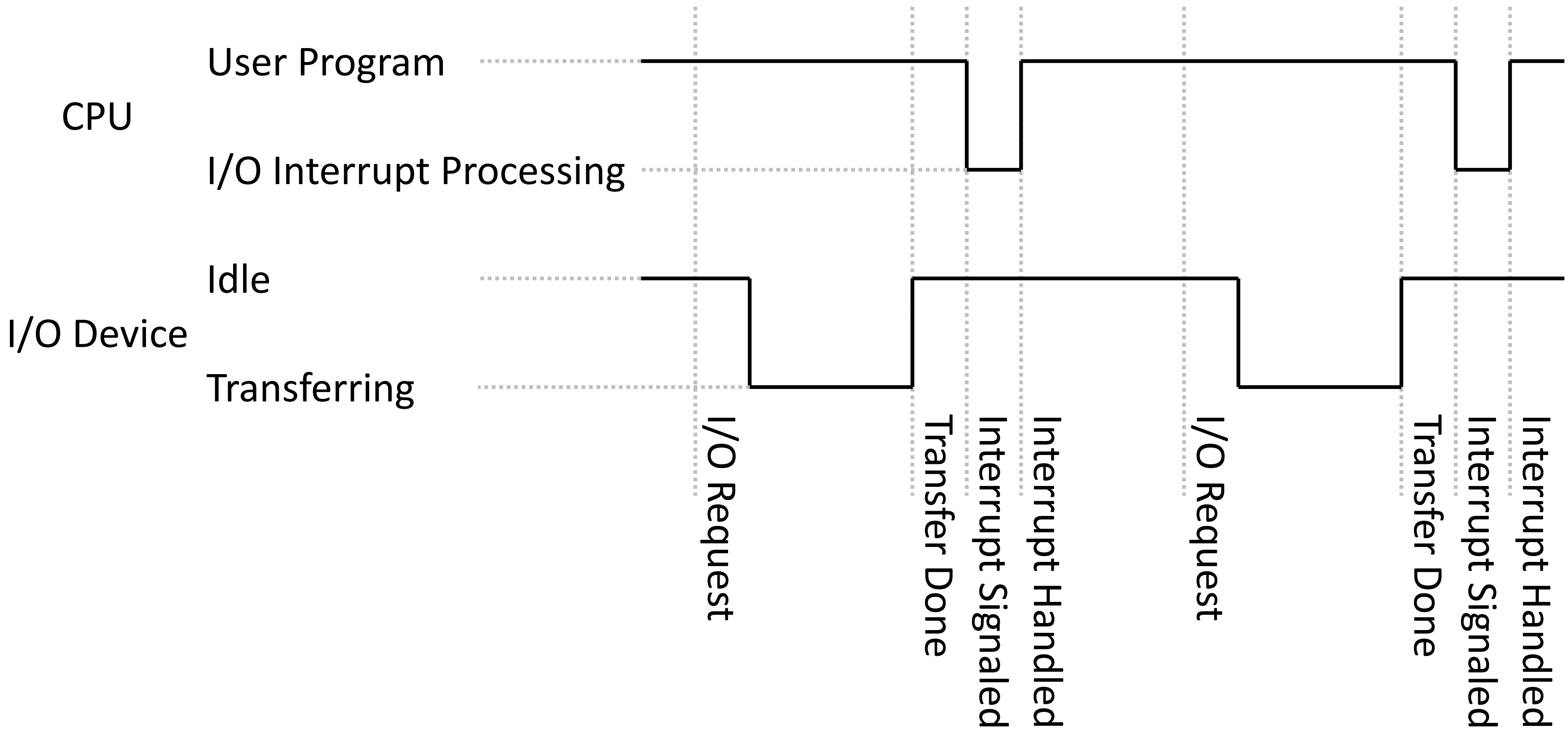
A Program Performing I/O

- The device driver loads the appropriate registers in the device controller
- The device controller
  - Examine the contents of the registers and determine what action to take
    - Example: "read a character from the keyboard"
  - Start the transfer of data from the device to its local buffer
  - **Inform the device driver once the transfer of data is complete**
- The device driver gives control to other parts of the operating system
  - Return the data or a pointer to the data (for a read operation)
  - Return status information (for other operations)
- How does the controller inform the device driver that it has finished its operation?

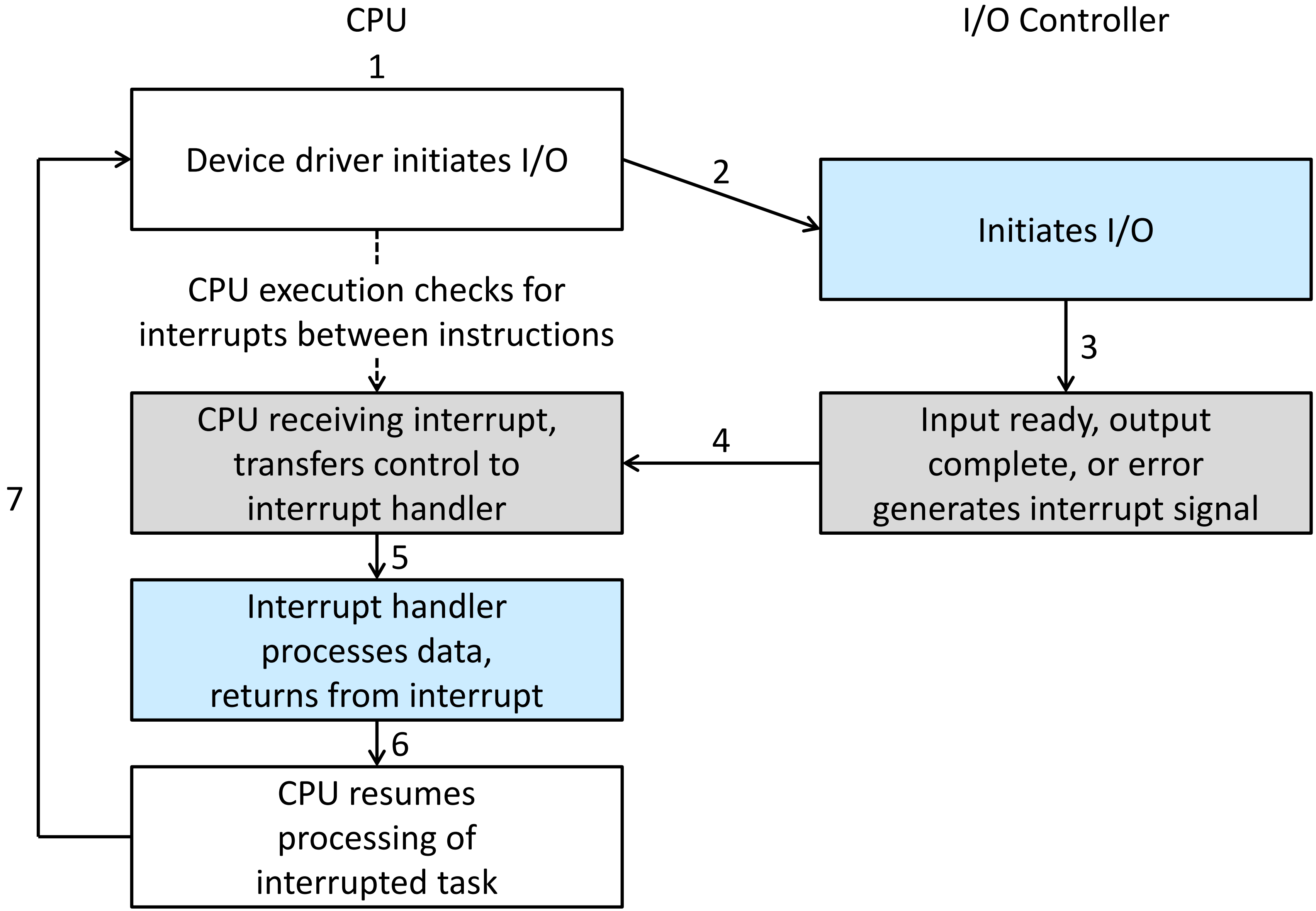
Overview of Interrupts

- **Hardware** may trigger an interrupt at any time by sending a signal to the CPU (usually by way of the system bus)
- Steps
  - When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location
  - The fixed location usually contains the starting address where the service routine for the interrupt is located
  - The interrupt service routine executes
  - On completion, the CPU resumes the interrupted computation
- Interrupts must be handled quickly, as they occur very frequently
  - **Interrupt vector**
    - An array (table) of pointers to interrupt routines

Interrupt Timeline



Interrupt-Driven I/O Cycle



CPU

I/O Controller

1

Device driver initiates I/O

2

Initiates I/O

CPU execution checks for interrupts between instructions

3

CPU receiving interrupt, transfers control to interrupt handler

4

Input ready, output complete, or error generates interrupt signal

7

5

Interrupt handler processes data, returns from interrupt

6

CPU resumes processing of interrupted task

Implementation of Interrupts

- **Interrupt-handler routine**
- Desired features
  - The ability to defer interrupt handling during critical processing
    - **Nonmaskable** and **maskable** **interrupt request lines**
  - An efficient way to dispatch to the proper interrupt handler for a device
    - Interrupt vector and **interrupt chaining**
  - Multilevel interrupts to distinguish between high- and low-priority interrupts and respond with the appropriate degree of urgency
    - **Interrupt priority levels**

Intel Processor Event-Vector Table

Vector Number	Description
0	Divide Error
1	Debug Exception
2	Null Interrupt
3	Breakpoint
4	INTO-Detected Overflow
5	Bound Range Exception
6	Invalid Opcode
7	Device Not Available
8	Double Fault
9	Coprocessor Segment Overrun (Reserved)
10	Invalid Task State Segment
11	Segment Not Present
12	Stack Fault
13	General Protection
14	Page Fault
15	(Intel Reserved, Do Not Use)
16	Floating-Point Error
17	Alignment Check
18	Machine Check
19--31	(Intel Reserved, Do Not Use)
32--255	Maskable Interrupts

Outline

- ❑ What Operating Systems Do
- ❑ **Computer-System Organization**
  - ➢ Interrupts, **Storage Structure**, I/O Structure
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

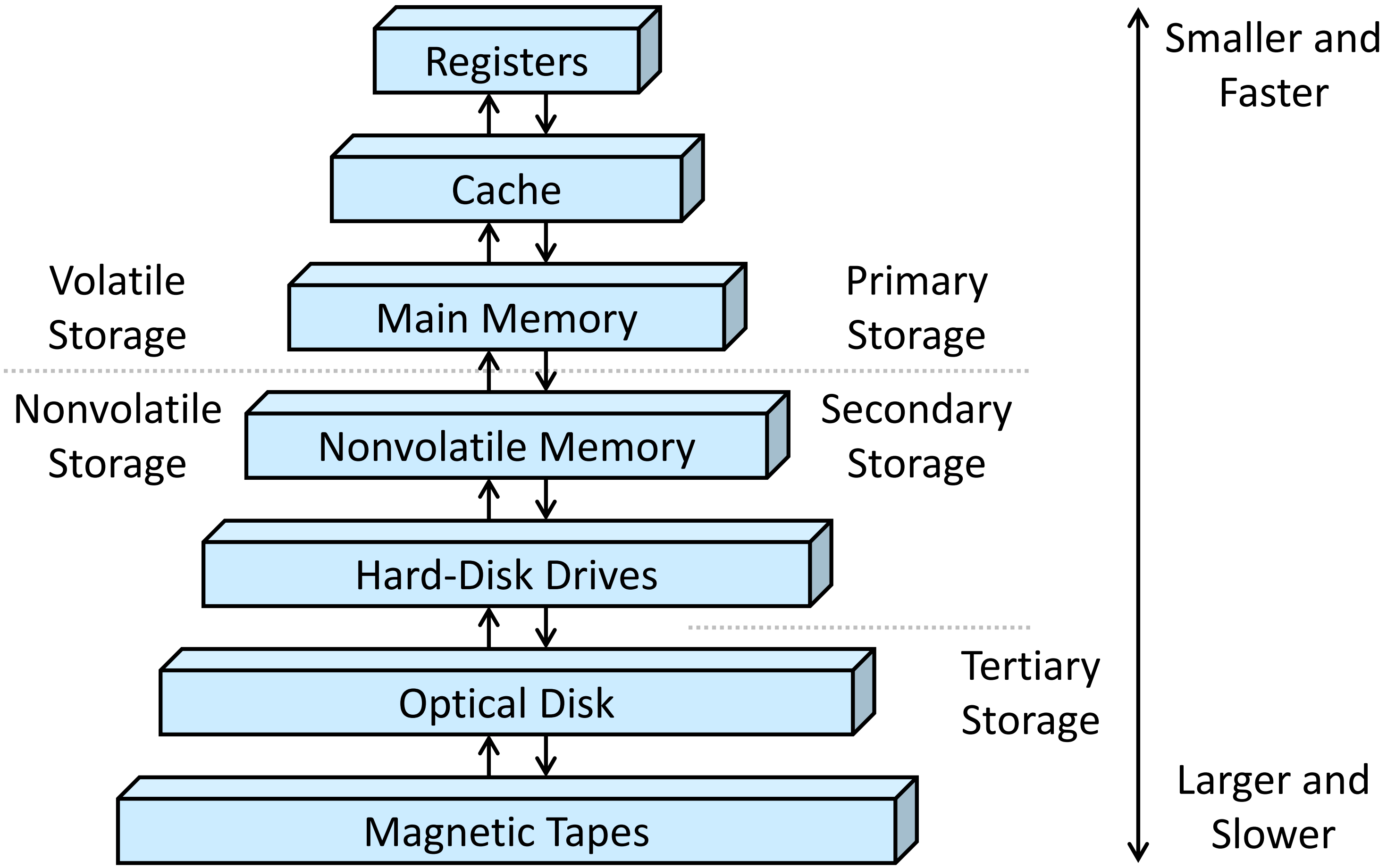
Main Memory

- ❑ General-purpose computers run most of their programs from rewritable memory
  - ➢ Called main memory, also called **random-access memory** (RAM)
  - ➢ Typically **dynamic random-access memory** (DRAM)
- ❑ The first program to run on computer power-on is a **bootstrap program** which then loads the operating system
  - ➢ Since RAM is **volatile**, the computer uses storage that is infrequently written to and is **nonvolatile**
  - ➢ Example: electrically erasable programmable read-only memory (EEPROM)
- ❑ Reference
  - ➢ von Neumann architecture

Secondary and Tertiary Storage

- The programs and data cannot reside in main memory permanently
  - Main memory is usually too small to store all needed programs and data
  - Main memory is volatile
- Secondary storage
  - **Hard disk drives** (HDD)
  - **Nonvolatile memory** (NVM) **devices**
- Tertiary storage
  - Examples: CD-ROM, blu-ray, magnetic tapes
- The main differences among the various storage systems lie in speed, size, and volatility

Storage-Device Hierarchy



Outline

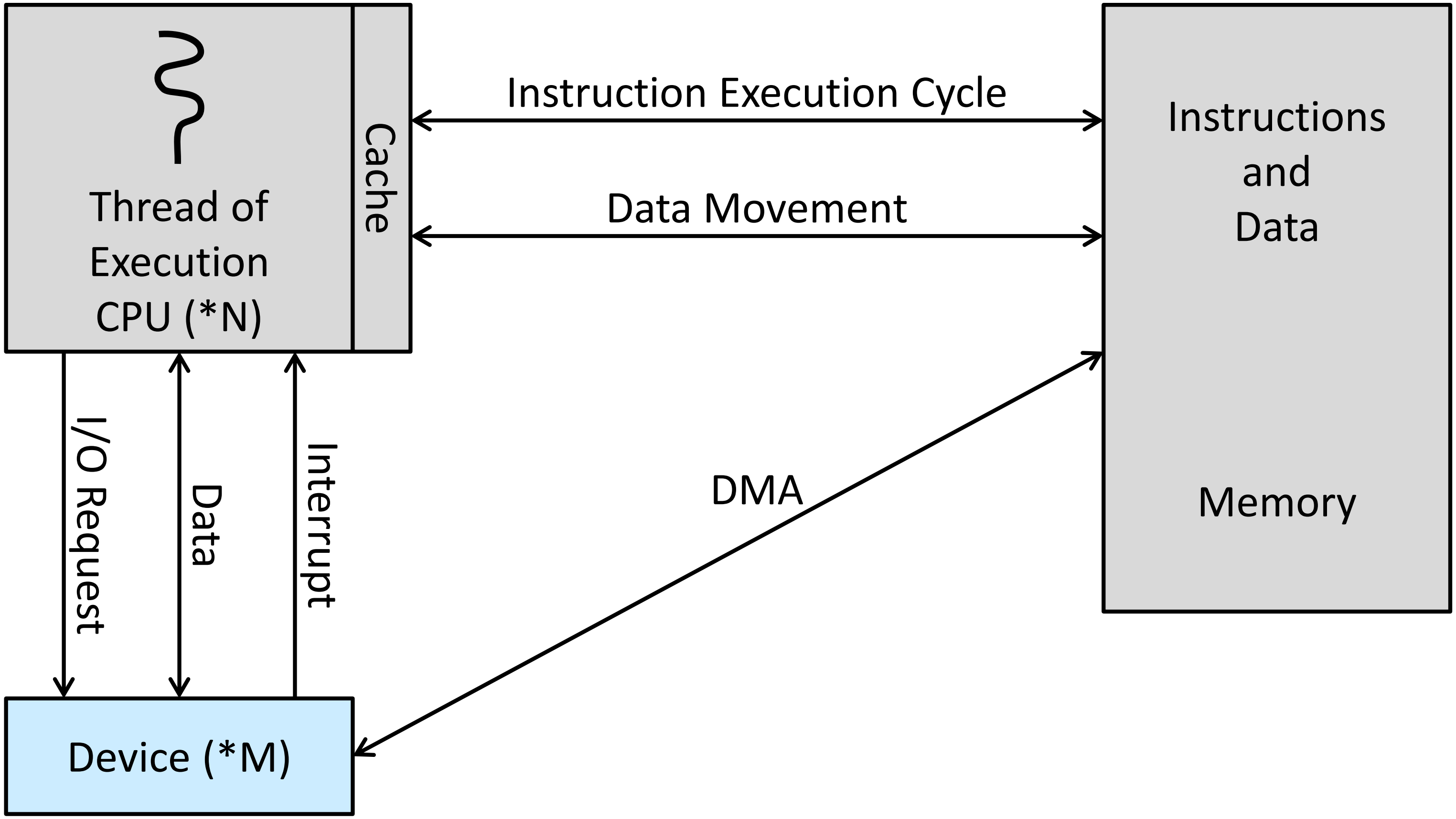
- ❑ What Operating Systems Do
- ❑ **Computer-System Organization**
  - ➢ Interrupts, Storage Structure, **I/O Structure**
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Direct Memory Access (1/2)

- ❑ A large portion of operating system code is dedicated to managing I/O
  - ➢ Interrupt-driven I/O is fine for moving small amounts of data, but it can produce high overhead for bulk data movement
- ❑ Direct memory access (DMA)
  - ➢ The device controller transfers an entire block of data directly to or from the device and main memory
    - No CPU intervention: the CPU is available to accomplish other work
  - ➢ Only one interrupt is generated per block to tell the device driver that the operation has completed
- ❑ Some high-end systems use switch (not bus) architecture
  - ➢ Multiple components can talk to others concurrently
  - ➢ DMA is even more effective in this case

Direct Memory Access (2/2)

- How a modern computer system works

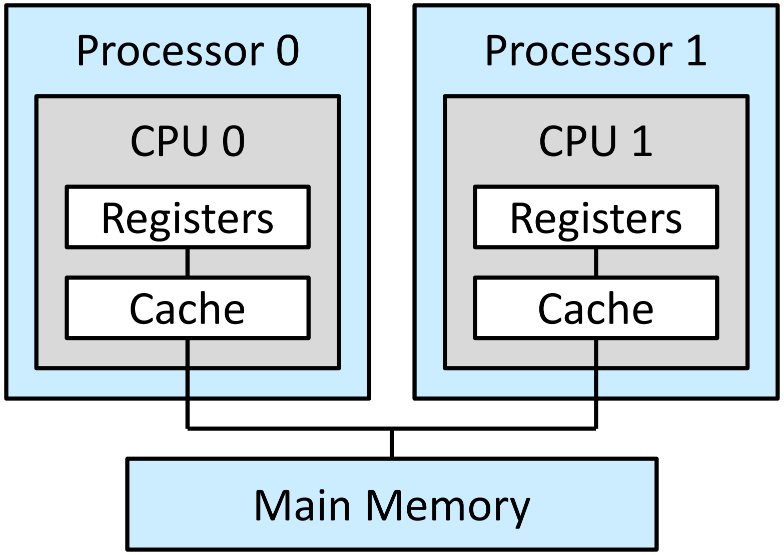


Outline

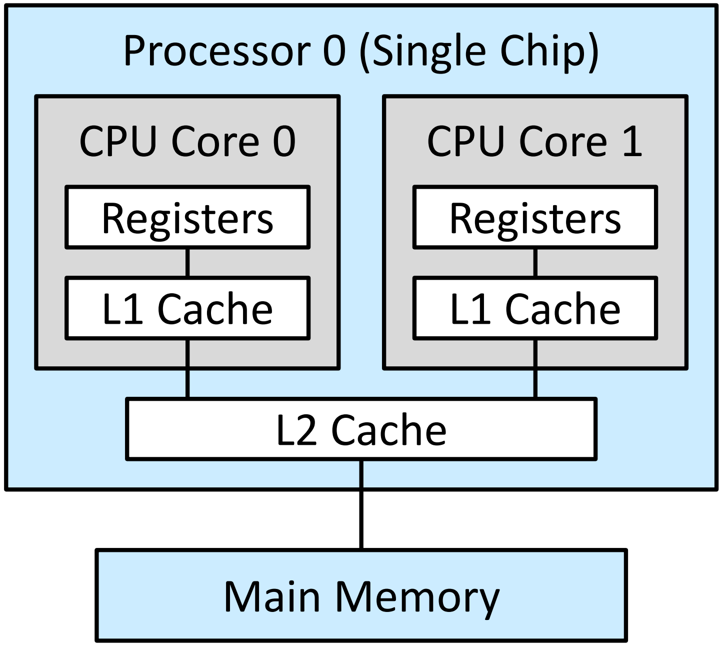
- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ **Computer-System Architecture**
- ❑ Operating-System Operations
- ❑ Resource Management
- ❑ Security and Protection
- ❑ Virtualization
- ❑ Distributed Systems
- ❑ Computing Environments
- ❑ Kernel Data Structures, Free and Open-Source Operating Systems

Computer-System Architecture

- ❑ Single-processor systems
- ❑ Multiprocessor systems
  - ➢ Growing in use and importance



Symmetric Multiprocessing Architecture



Dual-Core on the Same Chip

- ❑ Clustered systems
  - ➢ Two or more individual systems (or nodes) joined together

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ **Operating-System Operations**
  - ➢ Multiprogramming and Multitasking, Dual-Mode and Multimode Operation, Timer
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Operating-System Operations

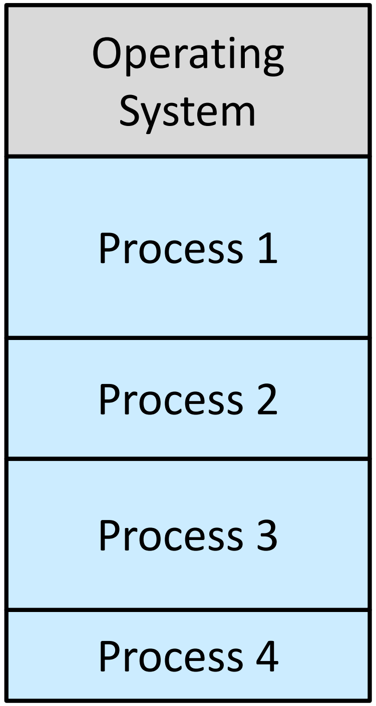
- Bootstrap program
  - Initialize all aspects of the system
  - Locate the operating-system kernel and load it into memory
- **System daemons**
  - Services provided outside of the kernel by system programs, runnning the entire time the kernel is running
- Events are almost always signaled by the occurrence of an interrupt
  - Hardware interrupt (described already)
  - Software interrupt (**exception** or **trap**)
    - Software error (e.g., division by zero)
    - A specific request from a user program that an operating-system service be performed by executing a special operation called a **system call**

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ **Operating-System Operations**
  - ➢ **Multiprogramming and Multitasking**, Dual-Mode and Multimode Operation, Timer
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Multiprogramming and Multitasking

- One of the most important aspects of operating systems is the ability to run multiple programs
  - A program in execution is termed a **process**
  - A set of processes is kept in memory
  - When a process has to wait, operating systems switch to another process
- **Multitasking** is a logical extension of multiprogramming
  - The CPU executes multiple processes by switching among them
  - The switches occur frequently and provide fast response times



Memory Layout

Multiprogramming and Multitasking

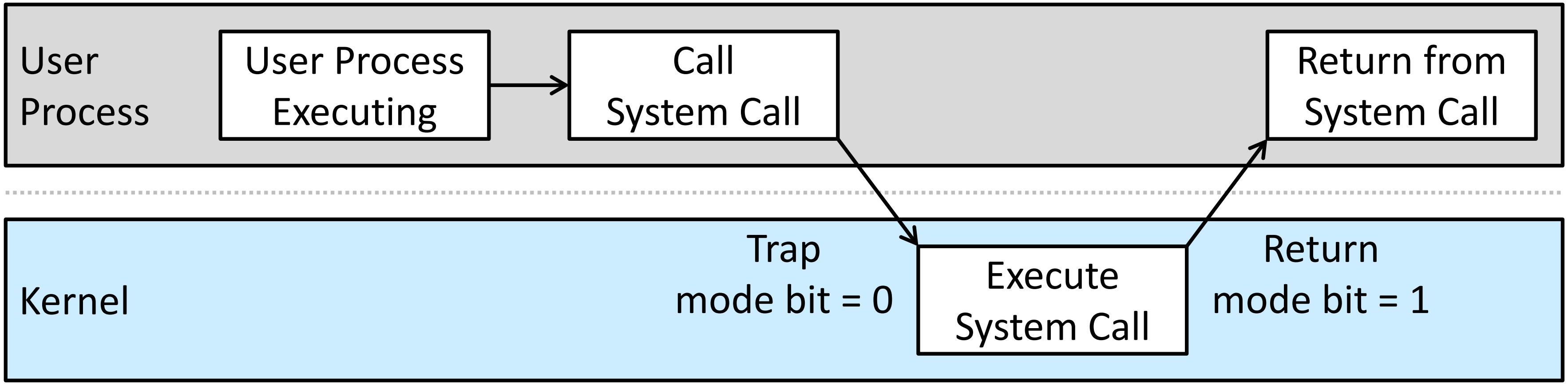
- Connections to following chapters
  - [9, 10] Having several processes in memory at the same time requires memory management
  - [5] The system must choose which process will run next
  - [13, 14, 15] Multiprogramming and multitasking systems must provide a file system
  - [11] The file system resides on a secondary storage which needs storage management
  - [17] A system must protect resources from inappropriate use
  - [6, 7] The system must ensure orderly execution for processes
  - [8] It may ensure that processes do not get stuck in a deadlock

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ **Operating-System Operations**
  - ➢ Multiprogramming and Multitasking, **Dual-Mode and Multimode Operation**, Timer
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Dual-Mode and Multimode Operation

- ❑ Distinguish between the execution of operating-system code and user-defined code
  - ➢ **User mode** and **kernel mode**
- ❑ A **mode bit** provided by hardware
  - ➢ A system call changes it to "kernel" and its return resets it to "user"



- ❑ Hardware allows **privileged instructions** to be executed only in kernel mode

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ **Operating-System Operations**
  - ➢ Multiprogramming and Multitasking, Dual-Mode and Multimode Operation, **Timer**
- ❑ Resource Management, Security and Protection, Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Timer

- We cannot allow a user program to
  - Get stuck in an infinite loop
  - Fail to call system services and never return control to the operating system
- A **timer** can be set to interrupt the computer after a specified period
  - Before turning over control to the user, the operating system ensures that the timer is set to interrupt

Outline

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- **Resource Management**
- Security and Protection
- Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- Free and Open-Source Operating Systems

Resource Management

- ❑ Process management
  - ➢ Chapters 3, 4, 5, 6, and 7
- ❑ Memory management
  - ➢ Chapters 9 and 10
- ❑ File-system management
  - ➢ Chapters 13, 14, and 15
- ❑ Mass-storage management
  - ➢ Chapter 11
- ❑ Cache management
- ❑ I/O system management
  - ➢ Chapter 12

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management
- ❑ **Security and Protection**
- ❑ Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Protection and Security

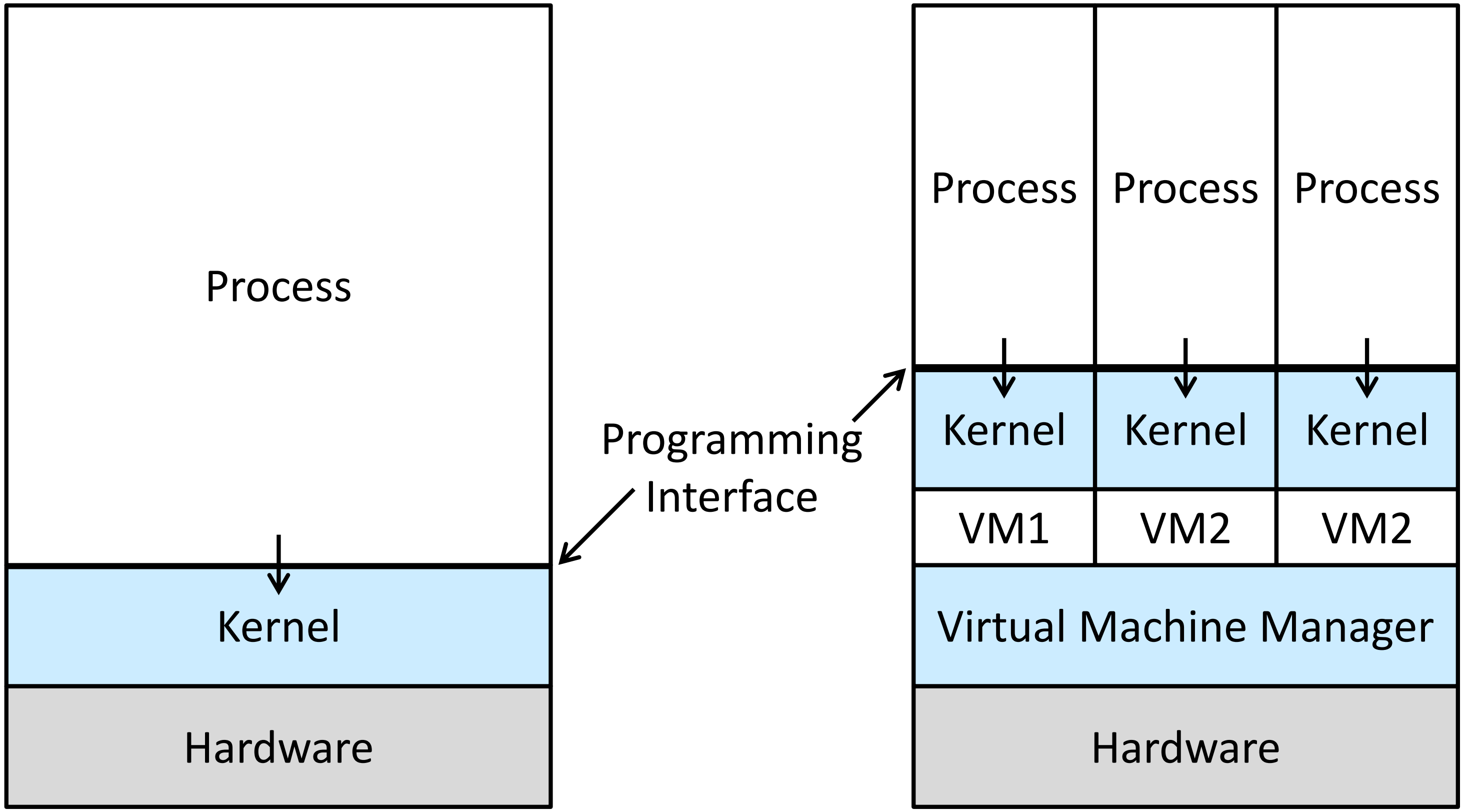
- **Protection**
  - Any mechanism for controlling the access of processes or users to the resources defined by a computer system
  - Chapter 17
- **Security**
  - Defend a system from external and internal attacks
    - Viruses and worms
    - Denial-of-service attacks
    - Identity theft
    - Theft of service
  - Chapter 16

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management
- ❑ Security and Protection
- ❑ **Virtualization**
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

Virtualization

- Abstract the hardware of a single computer into several different execution environments
  - Create the illusion that each separate environment is running on its own private computer



Outline

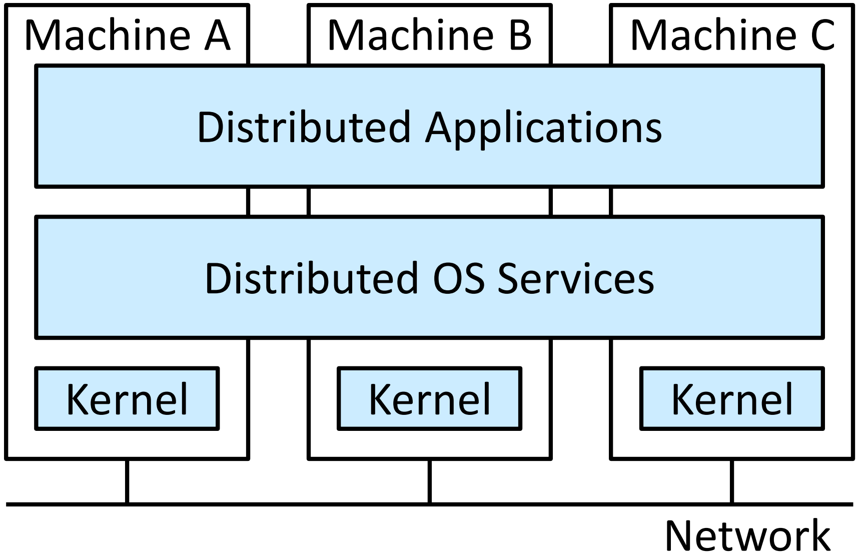
- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management
- ❑ Security and Protection
- ❑ Virtualization
- ❑ **Distributed Systems**
- ❑ Kernel Data Structures
- ❑ Computing Environments
- ❑ Free and Open-Source Operating Systems

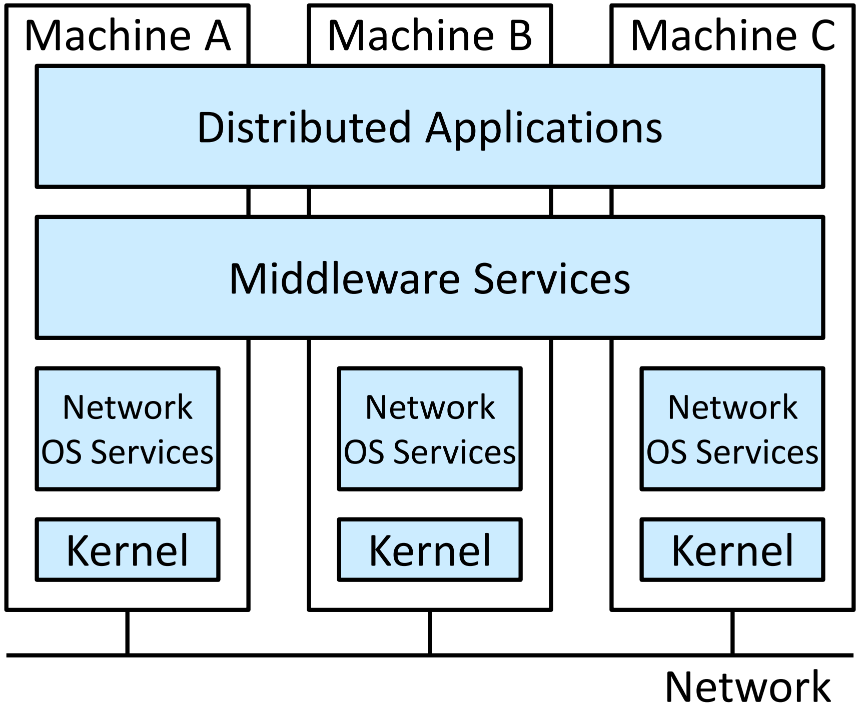
Distributed Systems

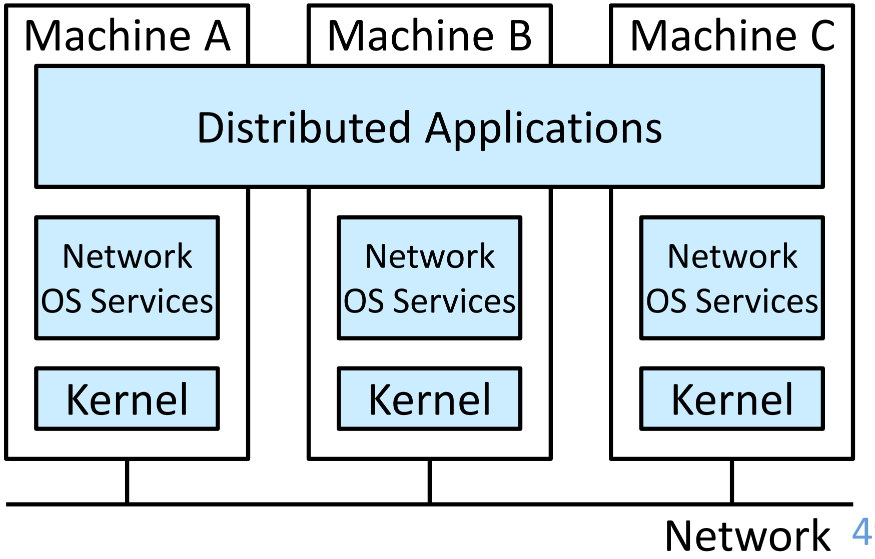
- A collection of networked, separate, and possibly heterogeneous computer systems
- A **network** is a communication path between two or more systems
  - **TCP/IP** is the most common network protocol
  - Personal/Local/Wide/Metropolitan Area Network (PAN/LAN/WAN/PAN)

Distributed Systems

- Distributed OSs →
- Network OSs ↘
- Middleware-based systems ↓







Outline

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed Systems
- **Kernel Data Structures**
- Computing Environments
- Free and Open-Source Operating Systems

Kernel Data Structures

- Lists, stacks, and queues
- Trees
- Hash functions and maps
- Bitmaps

Outline

- ❑ What Operating Systems Do
- ❑ Computer-System Organization
- ❑ Computer-System Architecture
- ❑ Operating-System Operations
- ❑ Resource Management
- ❑ Security and Protection
- ❑ Virtualization
- ❑ Distributed Systems
- ❑ Kernel Data Structures
- ❑ **Computing Environments**
- ❑ Free and Open-Source Operating Systems

Computing Environments

- Traditional computing
- Mobile computing
  - Apple iOS and Google Android
- Client-server computing
- Peer-to-peer computing
  - Not distinguish clients and servers
- Cloud computing
  - A logical extension of virtualization
  - Public cloud, private cloud, and hybrid cloud
  - Software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS)
- Real-time embedded systems

Computing Modes

- ❑ Batch mode for efficiency
  - ➢ Programs run independent of the environment
  - ➢ Results are written to files or output devices
  - ➢ Examples: compiler, machine learning
- ❑ Online mode for responsiveness
  - ➢ Users may interact with program
  - ➢ No time constraint is involved
  - ➢ Examples: spreadsheet, game, browsing
- ❑ Real-time mode for predictability
  - ➢ Programs interact strongly with the environment
  - ➢ Time constraints are imposed
  - ➢ Examples: autonomous vehicles, robot control

Outline

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- **Free and Open-Source Operating Systems**

Free and Open-Source OSs

- ❑ GNU/Linux
- ❑ BSD UNIX
- ❑ Solaris

Objectives

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments

Q&A