Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Computing Environments
- Open-Source Operating Systems

What is an Operating System?

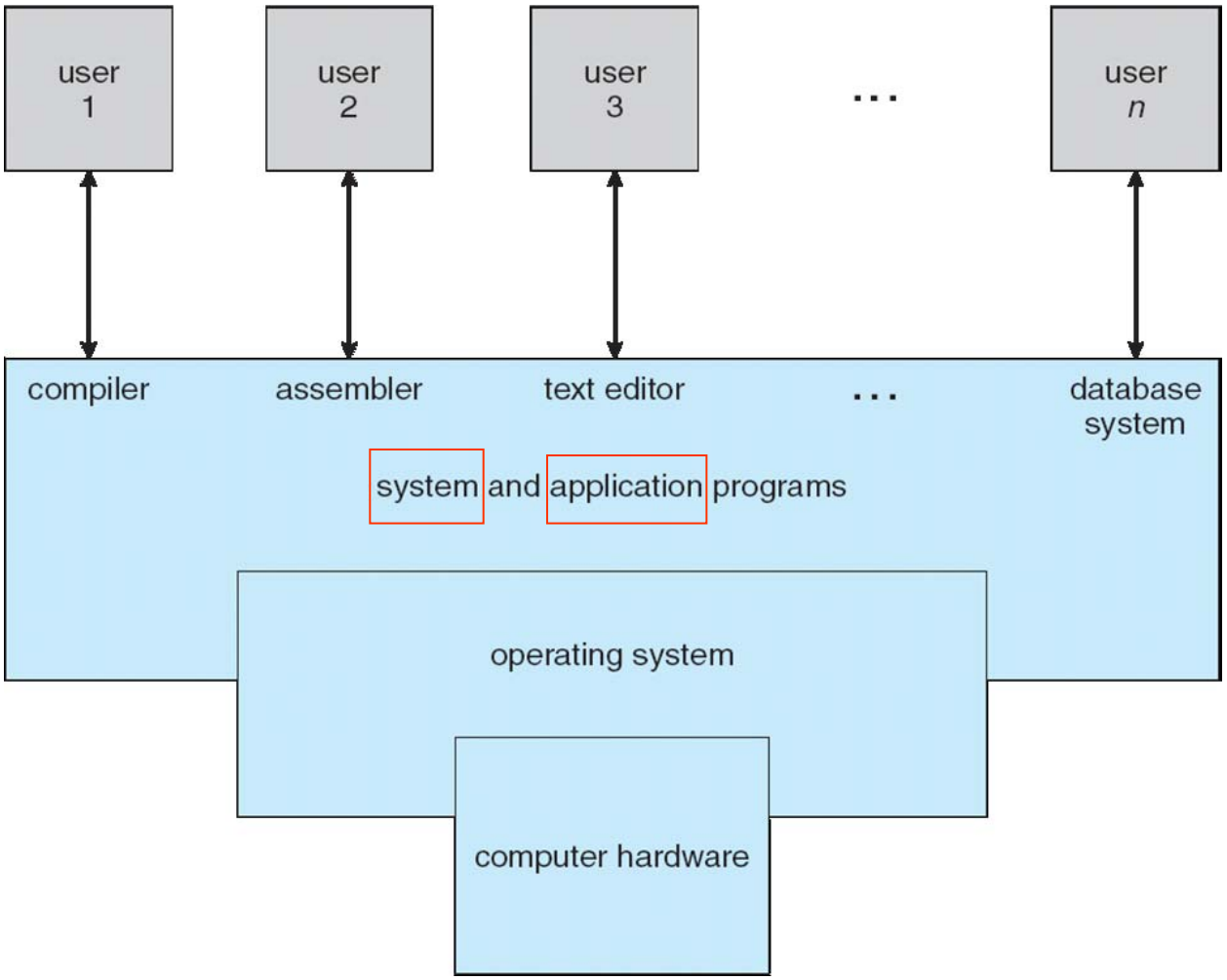
- A **program** that acts as an intermediary between a user of a computer and the computer hardware

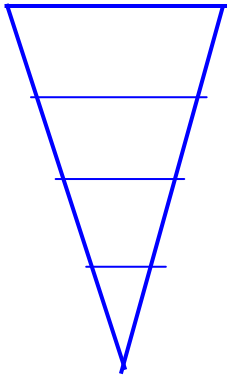
- **Operating system goals**:
  - (Execute user programs and) make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Abstract View of Computer System

- **Computer system** can be divided into four components
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs** – using the system resources to solve the computing problems of the users
    - Word processors, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

Four Components of a Computer System





Operating System Definition

- OS is a **resource allocator**
  - Manages all resources.
  - Decides between **conflicting requests** for efficient and fair resource use
- OS is a **control program**
  - Control the execution of user programs to **prevent errors** and **improper use** of the computer
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly, **Size**: 1MB ~ 1GB
- "The one program running at all times on the computer" is the **kernel program**.
  - Everything else is either a ***system program*** (ships with the operating system) or an ***application program***.

Operating System Views

User View:

- Users want convenience, **ease of use**
  - Don't care about resource utilization
- Users of **workstations** have **dedicated resources** but frequently use **shared resources** from **servers**
- **Personal computers**, **text console** => **GUI**

System View:

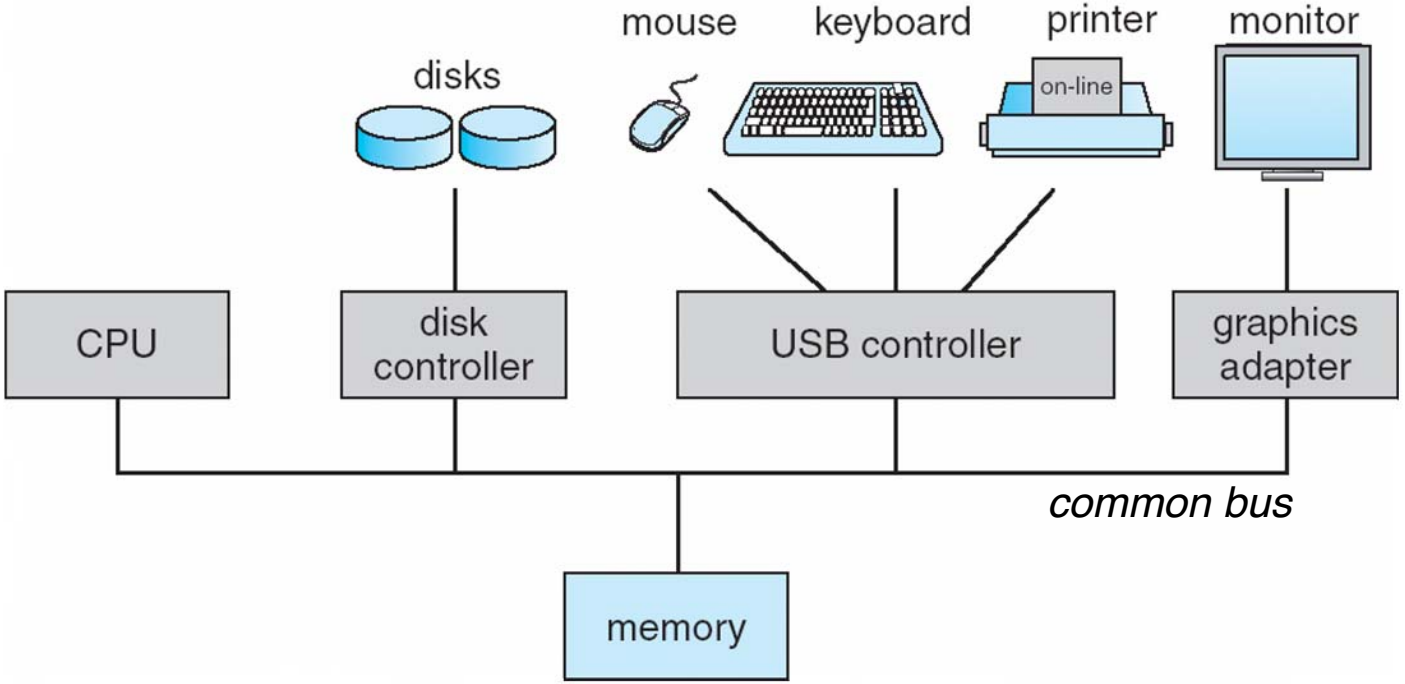
- **Mainframe** or **minicomputer** must keep all users happy
- **Handheld computers** (**smart phone**, **tablet**) are resource poor, optimized for usability and battery life
- Some computers have little or **no user interface**, such as **embedded computers** in **devices** and **automobiles**

Computer Startup

- **Bootstrap (起動) program**
  - Executed at **power-up** or **reboot**
  - Typically stored in **ROM** or **EPROM**, generally known as **firmware**
  - Initializes all aspects of system
    - check **keyboard**, **main memory**, **hard disk**.
  - Loads **operating system kernel** and starts execution

Computer System Organization

- **Computer-system operation (1)**
  - One or more **CPUs**, **device controllers** connect through **common bus** providing access to **shared memory**
  - ***Concurrent execution*** of **CPUs** and **devices** competing for **memory cycles**



Computer-System Operation

- **Computer-system operation (2)**
  - **I/O** devices and the **CPU** can execute **concurrently**
  - Each **device controller** is in charge of a particular **device type**
  - Each **device controller** has a **local buffer**

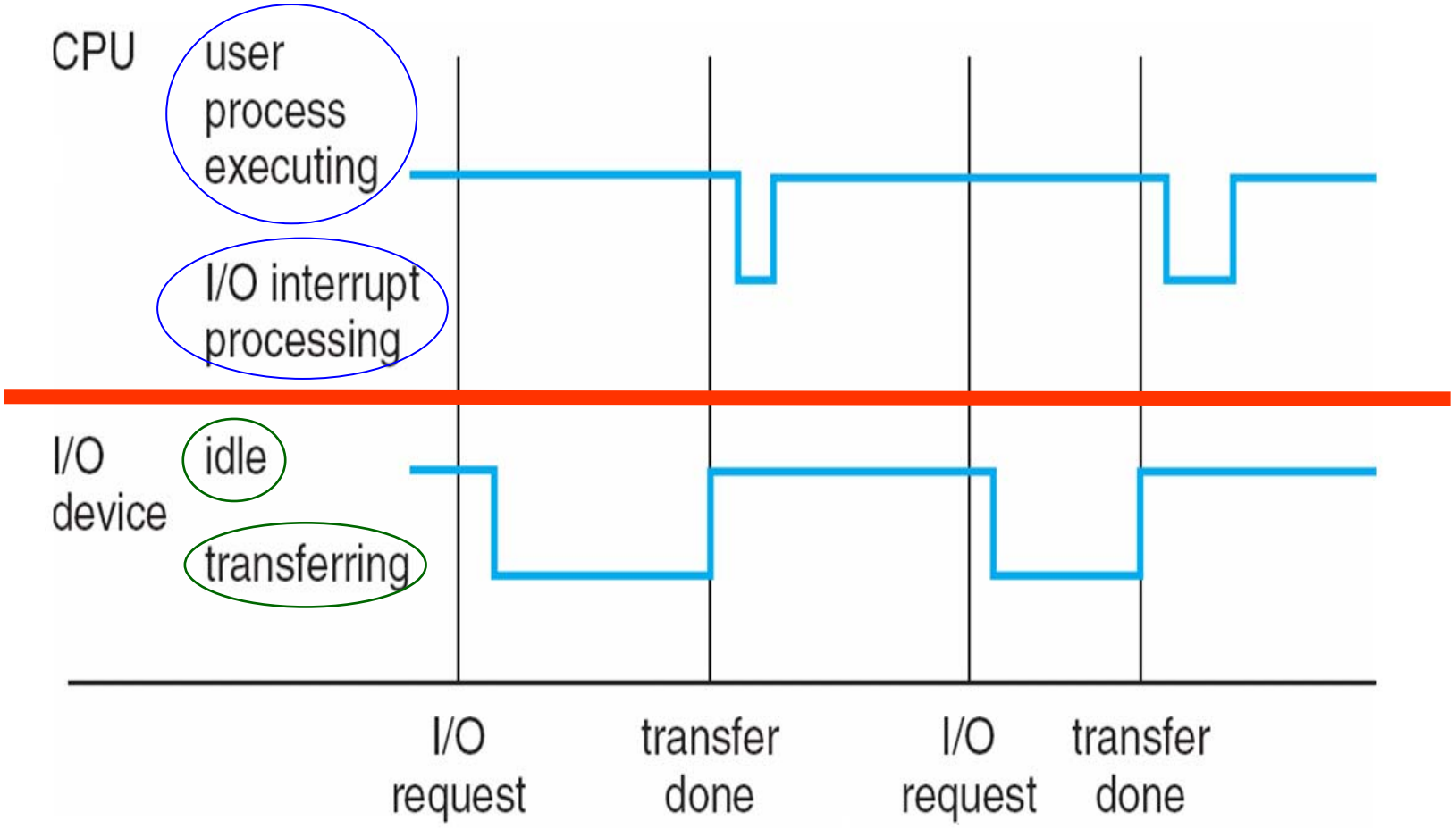
    I/O devices: keyboard, mouse, printer, hard disk, …

  - **Device controller** informs **CPU** that it has finished its operation by causing an ***interrupt***
  - **CPU** moves data from/to main memory to/from local buffers
  - **I/O** is from the device to local buffer of controller

Interrupt Concepts and Handling

- **Interrupt** transfers control to the **interrupt service routine** generally, through the **interrupt vector**, which contains the addresses of all the service routines
- **Interrupt architecture** must save the **address** of the interrupted instruction
- **Incoming interrupts** may be ***disabled*** while another interrupt is being processed to prevent a ***lost interrupt***
- A **trap** is **a *software-generated interrupt*** caused either by an **error** or a **user request (system call)**
- The **operating system** preserves the state of the **CPU** by storing **registers** and the **program counter (PC)**
- An operating system is ***interrupt driven***

Interrupt Timeline



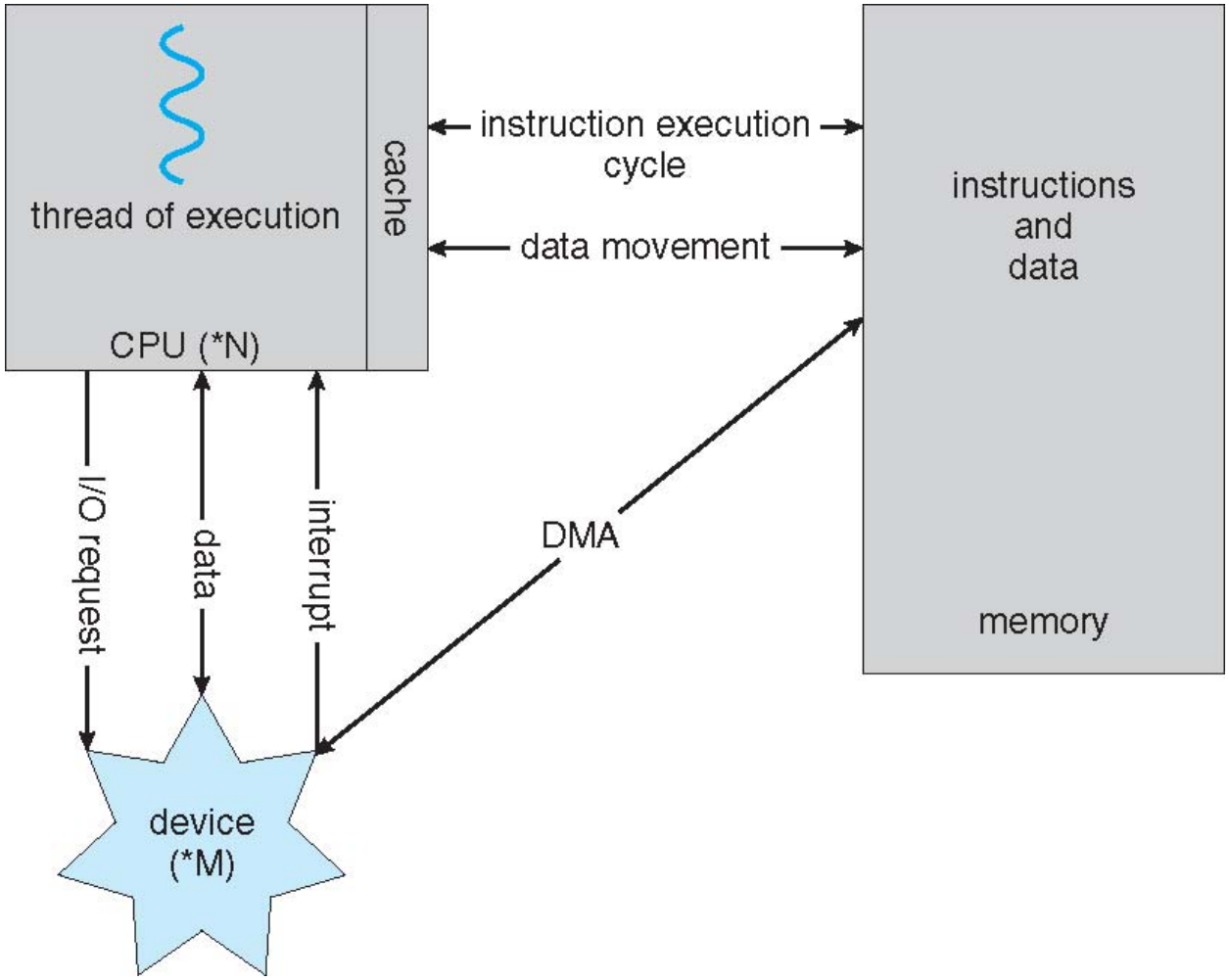
I/O Structure

- **Synchronous I/O (e.g. read the keyboard)**
  - After I/O starts, control returns to user program only upon I/O completion. **No** simultaneous I/O processing
  - **System call** – request to the OS to allow a user program to wait for I/O completion
- **Asynchronous I/O**
  - After I/O starts, control returns to user program without waiting for I/O completion. **Can do** simultaneous I/O processing
  - **System call** – request to the OS to allow a user program to not wait for I/O completion
- **I/O waiting**
  - **Interrupt**, CPU needs no waiting (modern CPU)
  - **Polling**, CPU enter a wait loop and check the status of device

Direct Memory Access Structure

- Only **one interrupt** is generated per **block** **(block I/O)**, rather than the **one interrupt per byte** (**character I/O**)
- Used for **high-speed I/O devices** able to transmit information at close to **memory speeds**
- **Device controller** transfers blocks of data from **buffer storage** directly to **main memory** without **CPU intervention**

DMA for Block I/O



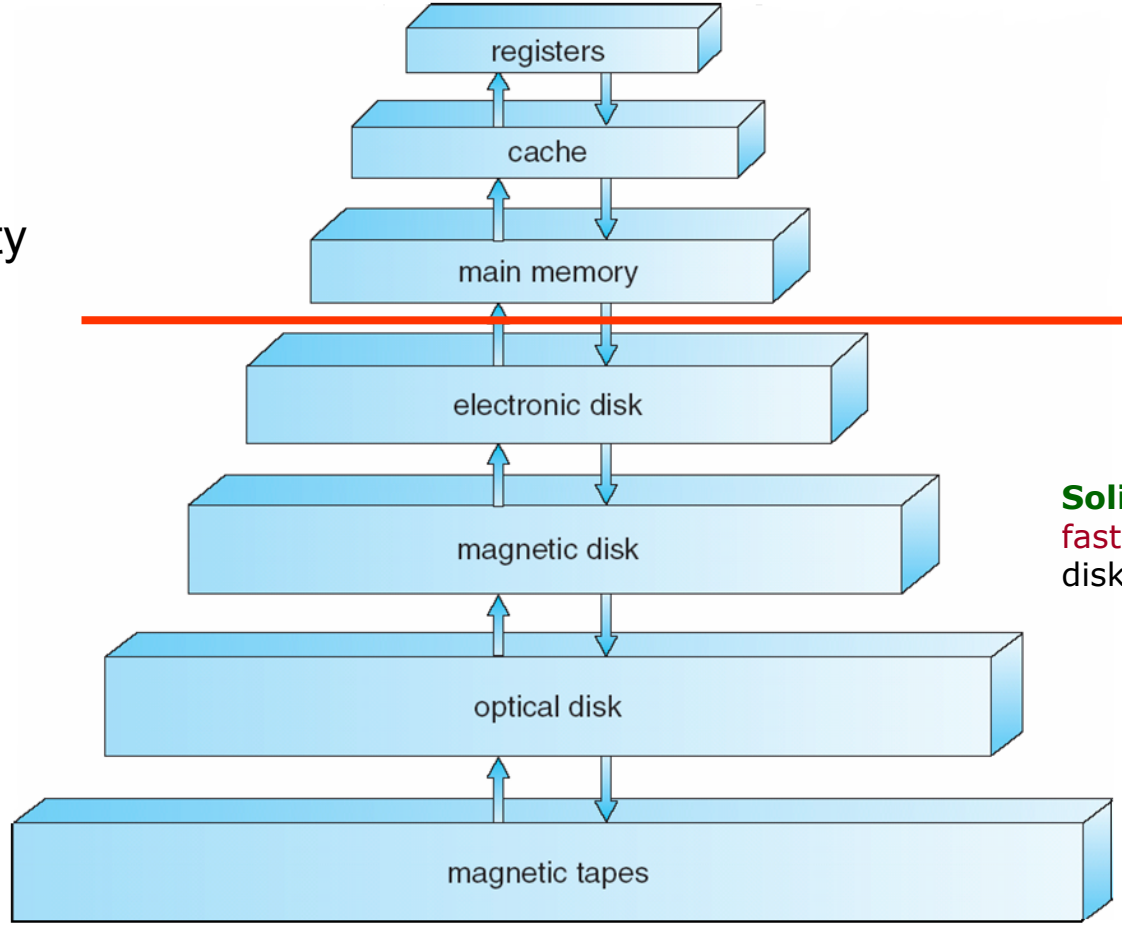
Storage-Device Hierarchy

- Speed
- Cost
- Volatility

tracks

sectors

clusters



Solid-state disks : faster than magnetic disks, nonvolatile

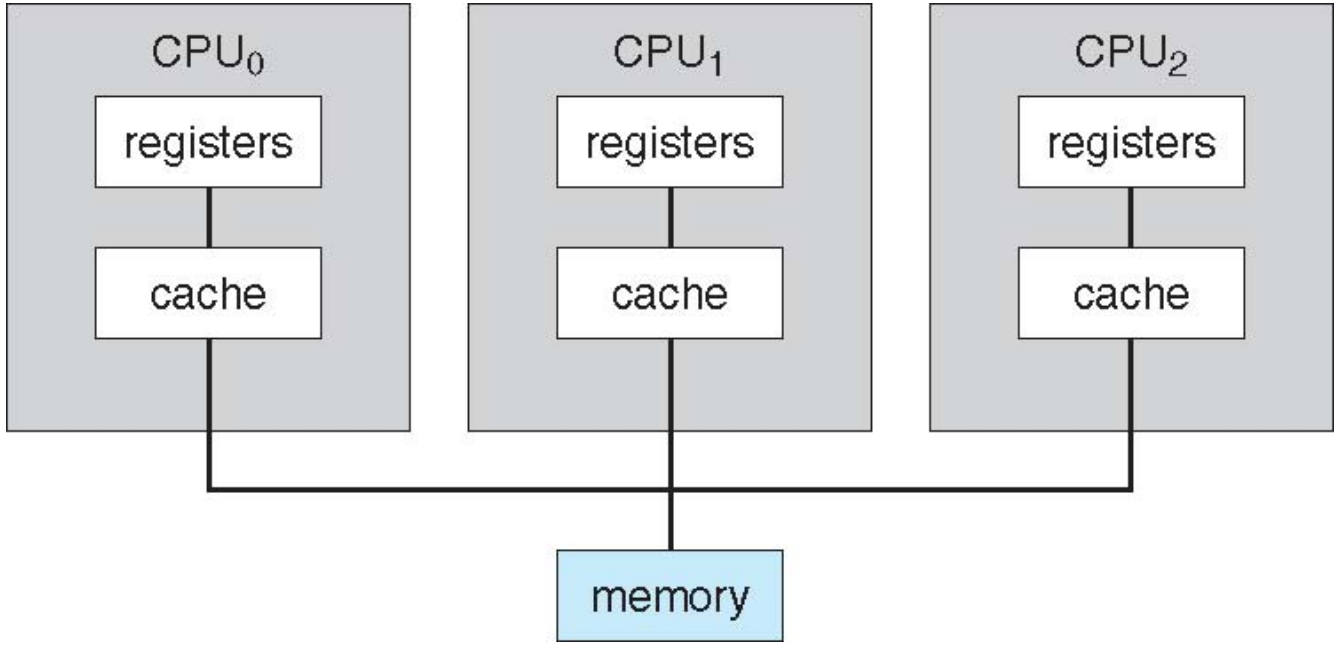
Caching (暫存)

- Important principle, performed at many levels in a computer (in **hardware**, **operating system**, **software**)
- Copy **Information** in use from **slower** to **faster** storage temporarily
- **Faster storage** (cache) **checked first** to determine if information is there
  - **If it is**, information used directly from the cache (fast)
  - **If not**, data copied to cache and used there
- **Cache** (暫存區) smaller than storage being cached
  - **Cache management**: important design problem
  - Cache **size** and **replacement policy**

Computer-System Architecture

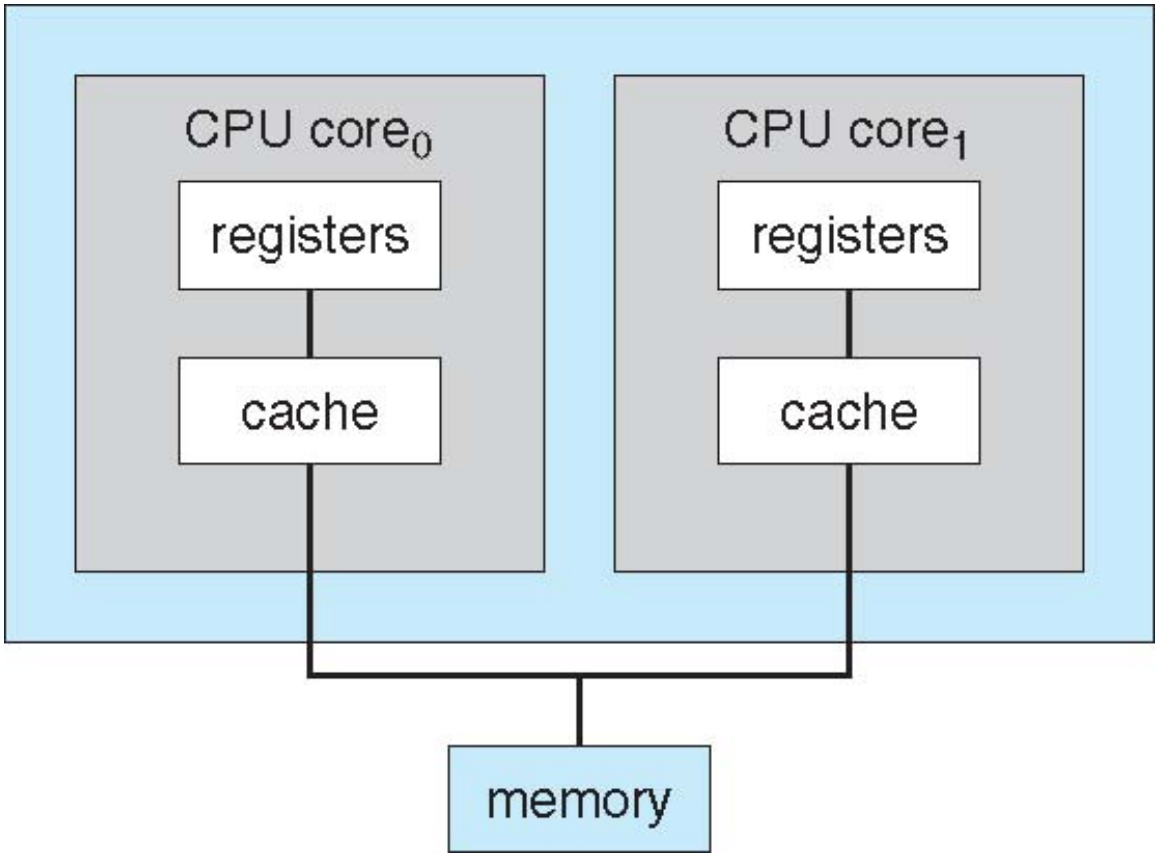
- Most systems use a **single general-purpose** processor (PDAs through mainframes)
  - Most systems have **special-purpose** processors as well (e.g. GPU)
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types in OS design
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

Symmetric Multiprocessing Architecture



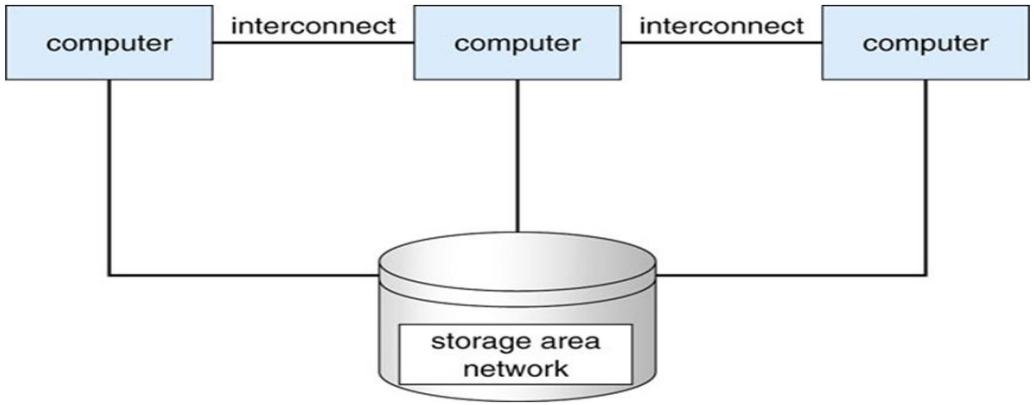
- **UMA** (uniform memory access) and **NUMA** architecture
- Multi-chip and **multicore**

A Dual-Core Design



Clustered Systems (叢集系統)

- Like **multiprocessor systems**, but **multiple systems** (**OS**) working together
  - Usually sharing **storage** (**hard disk**) via a storage-area network (SAN, Fig. 1.8)
  - Provides a **high-availability service** which **survives failures**
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use parallelization



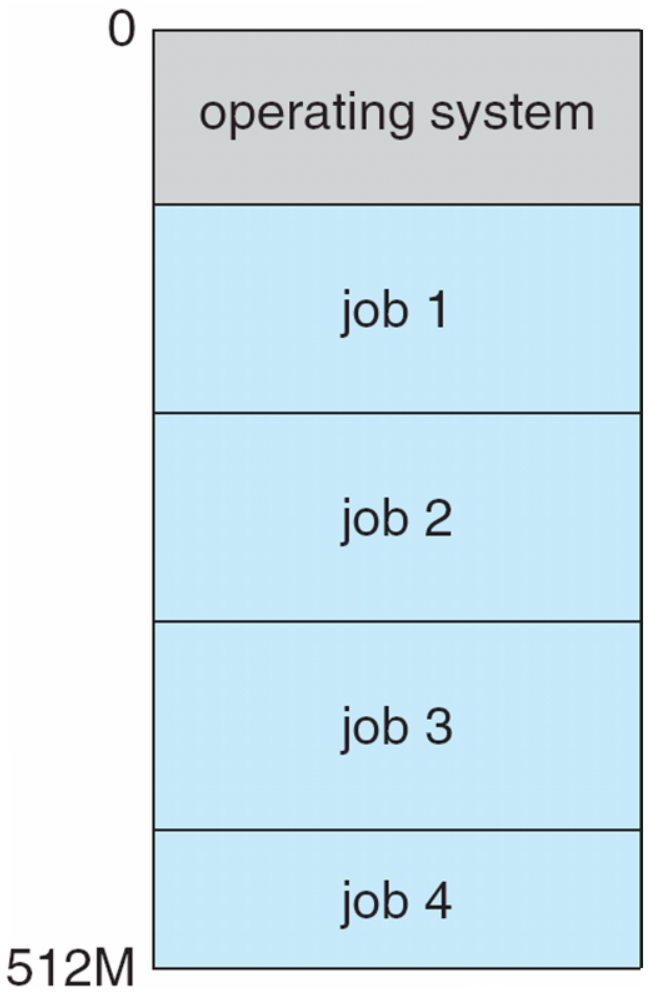
Operating System Types

- **Batch system**
  - jobs are **queued** and wait for **CPU** to execute one at a time
  - **CPU** is **idled** when waiting **I/O** to complete

- **Multiprogramming (多工) OS** is needed for efficiency
  - **Multiprogramming OS** organizes jobs (code and data) so **CPU** always has one to execute
  - **One job selected and run** via **job scheduling**
  - When a job has to **wait** (e.g. I/O), **OS switches to another job**

- **Timesharing (分時多工, multitasking) OS** is a logical extension (=> **multiuser**)
  - **CPU** switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - **Each user** has at least one program executing in memory ⇨**process manage**
  - If **several jobs ready** to run at the same time ⇨ **CPU scheduling**
  - If processes **don't fit in memory**, **swapping** moves them in and out to run

Memory Layout for Multiprogrammed System



Memory management

CPU scheduling

Single-user OS

(e.g. MS Windows)

Multi-user OS

(e.g. Linux)

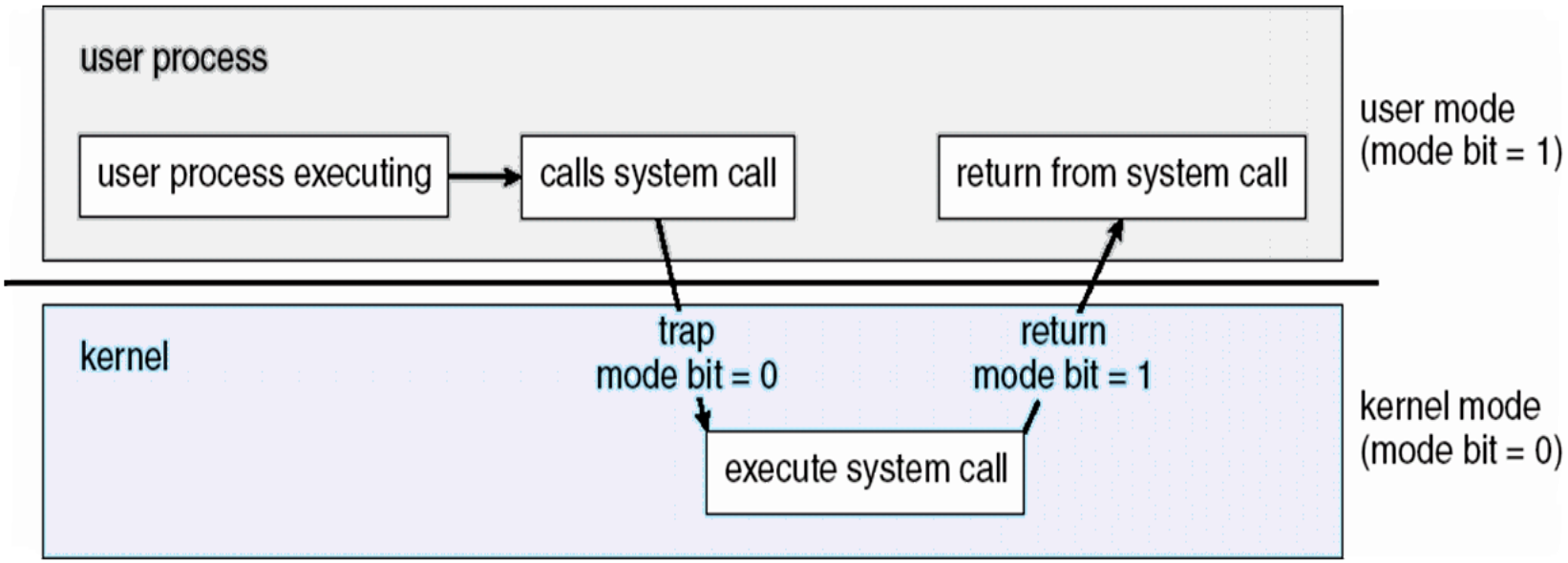
Operating-System Operations

- **Interrupt** driven (interrupts generated by hardware devices)
- **Software error or request** creates **exception** or **trap** **(可允許)**
  - **division by zero**, s**ystem call** (request for OS service)
- **Other process problems** include **infinite loop**, **illegal instruction**, **illegal memory access** (access other program's memory) **(不能允許)**
  - require **protection** (CPU and memory protection)

- **Dual-mode** operation allows **OS** to protect itself and other system components **(I/O devices)** (I/O device protection)
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - to **distinguish** when system is running **user code** or **kernel code**
    - Some **privileged instructions**, only executable in **kernel** mode
    - **System call** changes mode to **kernel**, return from call resets it to **user**
  - Intel **8088 CPU** without **mode bit** => **MS-DOS** without protection

Transition from User to Kernel Mode

- **Timer** to prevent **infinite loop** / process hogging resources **(protect CPU)**
  - **OS** set up a hardware counter before scheduling a process to regain **CPU**
  - When **counter come to zero**, an **interrupt** is generated => Enter **OS**



Process Management - Process

- A **process** is a program in execution.
  - **Program** is a ***passive*** *entity*, **process** is an ***active*** *entity*.
- **Process** needs **resources** to accomplish its task
  - **CPU**, **memory**, **I/O**, **files**, Initialization data
- **Process** termination requires **reclaim** of any reusable resources

- **Single-threaded** process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- **Multi-threaded** process has **one program counter per thread**

- Typically system has **many processes**, some user, some operating system **running concurrently** on one or more CPUs
  - **Concurrency** by multiplexing the CPUs among the processes / threads

Process Management – OS Activities

The **operating system** is responsible for the following activities in connection with **process management**:

- **Creating** and **deleting** both user and system processes
- **Suspending** and **resuming** processes
- Providing mechanisms for **process synchronization**
- Providing mechanisms for **process communication**
- Providing mechanisms for **deadlock handling**

Memory Management

- All **data** in memory before and after processing
- All **instructions** in memory in order to execute
- **Memory management** determines **what** is in memory and **when**

  => Optimizing **CPU utilization** and computer response to users

- **Memory management activities**
  - **Keeping track** of **which parts** of memory are currently being used and by **whom**
  - Deciding which **processes** (or parts thereof) and **data** to **move into and out** of memory
  - **Allocating** and **deallocating** memory space as needed

File Management

- **OS** provides **uniform**, **logical** view of **information storage**
  - **Storage devices**, e.g. **disk** drive, **tape** drive
    - **Varying properties** include **access speed**, **capacity**, **data-transfer rate**, **access method** (**sequential or random**)
  - Abstracts physical properties to logical storage unit -- **file**
  - Files usually organized into **directories**
  - **Access control** on most systems to determine who can access what

- **OS activities** include
  - Creating and deleting files and directories
  - Primitives to manipulate files and dirs
  - Mapping files onto secondary storage
  - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually **disks** used to store data that does **not fit in main** memory or data that must be **kept for a long period of time**
- Entire **speed of computer operation** hinges on **disk subsystem** and its algorithms

- **OS activities**
  - **Free-space** management
  - Storage allocation
  - Disk scheduling

- **Some storage need not be fast**
  - Tertiary (第三級) storage includes **optical storage**, **magnetic tape**
  - Varies between **WORM** (write-once, read-many-times) and **RW** (read-write)
  - Still must be managed

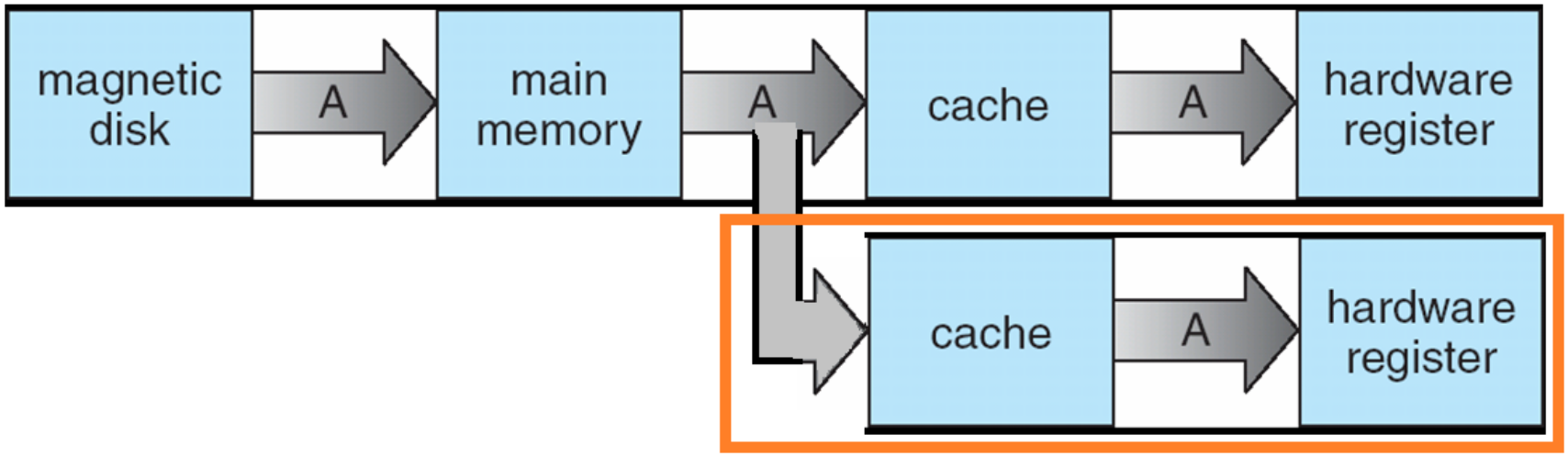
Various Levels of Storage

- Movement between levels of **storage hierarchy** can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Caching: Migration A from Disk to Register

- **Multitasking environments** must be careful to use **most recent value**, no matter where it is stored in the **storage hierarchy**



- **Multiprocessor environment** must provide **cache coherency** in hardware such that all **CPUs** have the most **recent value** in their **cache**
- **Distributed environment** situation even more complex
  - Several copies of a datum can exist

Kernel I/O Subsystem

- One purpose of **OS** is to
  - **hide** peculiarities of **hardware devices** from the **user**
- **I/O subsystem** **responsible** for
  - I/O scheduling (scheduling the I/O requests), **disk r/w requests**
  - buffering (storing data temporarily before being transferred), **write file**
  - caching (storing data in faster storage for performance), **disk cache**
  - spooling (overlapping output of one job with input of other jobs), **printing**
  - General **device-driver interface**
  - **Drivers** for specific **hardware devices**

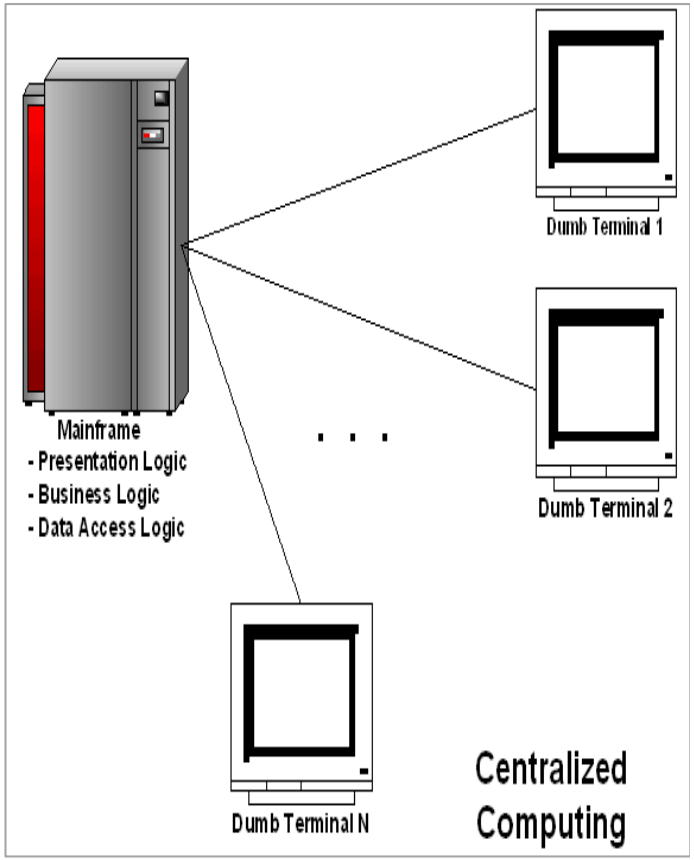
Protection and Security

- **Protection** – any mechanism for **controlling access** of **processes** or **users** to **resources** defined by the OS
- **Security** – **defense** of the system against **internal** and **external attacks**
  - Huge range, including **denial-of-service**, **worms**, **viruses**, **identity theft**, theft of service

- **OS** first distinguish among users, to determine **who** can do **what**
  - **User identity** (**UID**, security ID) include **user name** and associated **number**, one per user
  - **UID** then associated with all **files**, **processes** of that user to determine access control
  - **Group identifier** (**GID**) allows **set of users** to be defined. It is also associated with each **process**, **file**
  - **Privilege escalation** (提升) allows user to change to **effective ID** (**set UID**) with more rights

Computing Environments - Traditional

- **Stand-alone general purpose machines**
  - But blurred as most systems **interconnect** with others (i.e. **the Internet**)
- **Portals** provide web access to internal systems
- **Network computers** are like **web terminals**
- **Mobile computers** interconnect via **wireless networks**
- **Networking** becoming ubiquitous –
  - even home systems use **firewalls** to protect home computers from Internet attacks



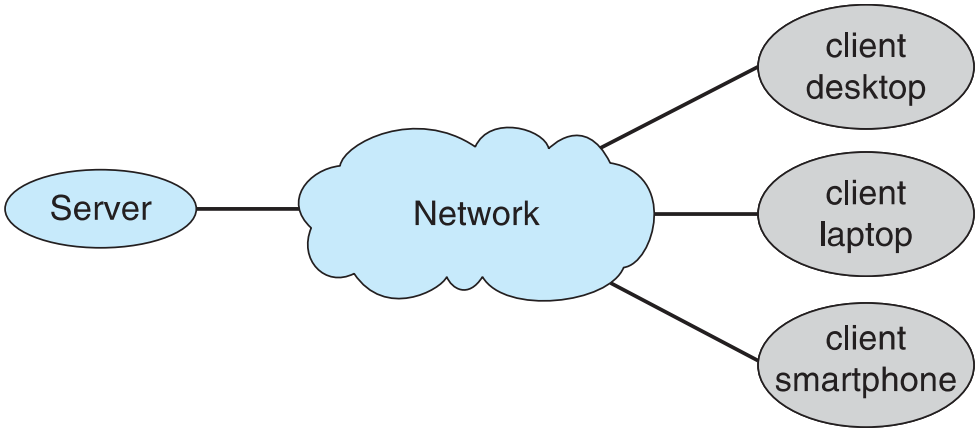
Computing Environments - Mobile

- **Handheld smartphones**, **tablets**, etc
- What is the **functional difference** between them and a **traditional laptop**?
  - **Extra feature** – more OS features: **GPS**, **gyroscope** (陀螺儀)
  - Allows new types of **apps** like ***augmented reality***
  - Use **IEEE 802.11 wireless**, or **cellular (3G or 4G) data networks** for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Client-Server

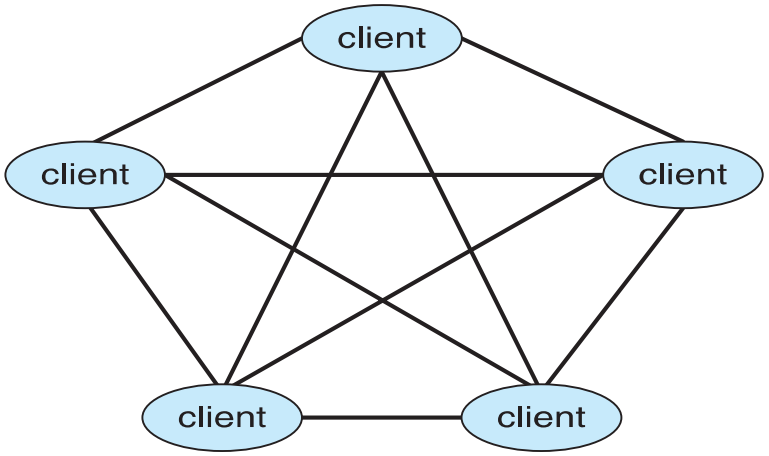
- **Client-Server Computing**
  - **Dumb terminals** replaced by **smart PCs**

- Many systems now **servers**, responding to requests generated by **clients**
  - **Compute-server system** provides an **interface** to **client** to request **services** (e.g. database)
  - **File-server system** provides **interface** for **clients** to store and retrieve **files**



Computing Environments - Peer-to-Peer

- Another model of **distributed system**
- **P2P** does **not distinguish clients and servers**
  - Instead **all nodes** are considered **peers**
  - May each act as client, server or both
  - Node must join **P2P network**
    - Registers its service with **central lookup service** on network, or
    - Broadcast request for service and respond to requests for service via **discovery protocol**
  - Examples include ***Napster*** and ***Gnutella***, **Voice over IP (VoIP)** such as ***Skype*** or ***Line***

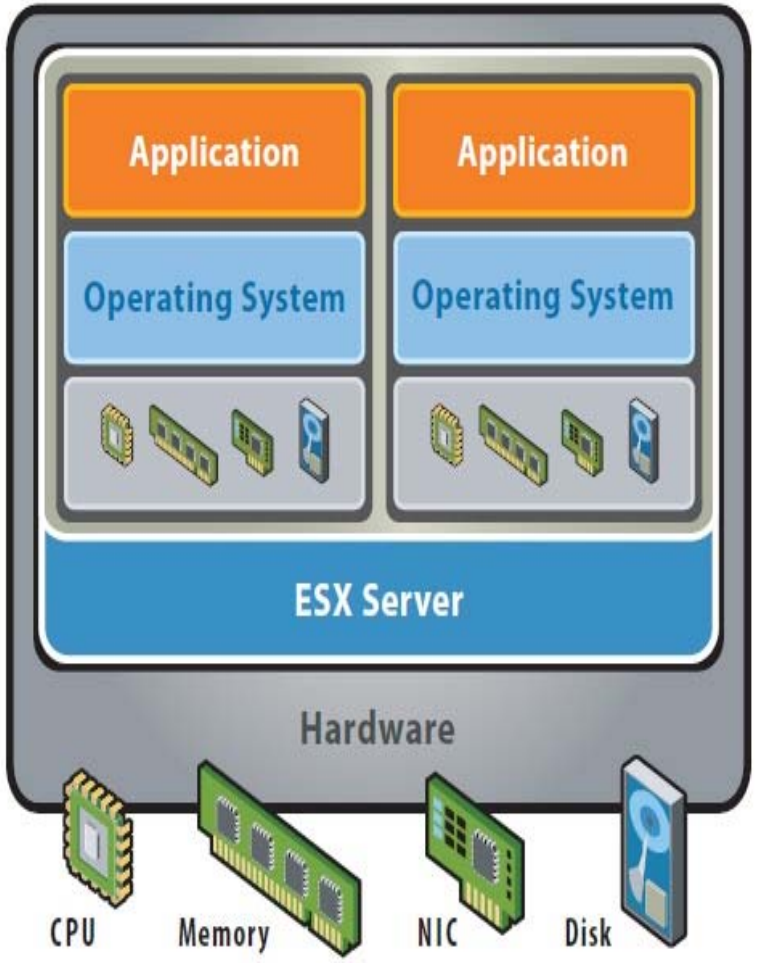


Computing Environments - Virtualization

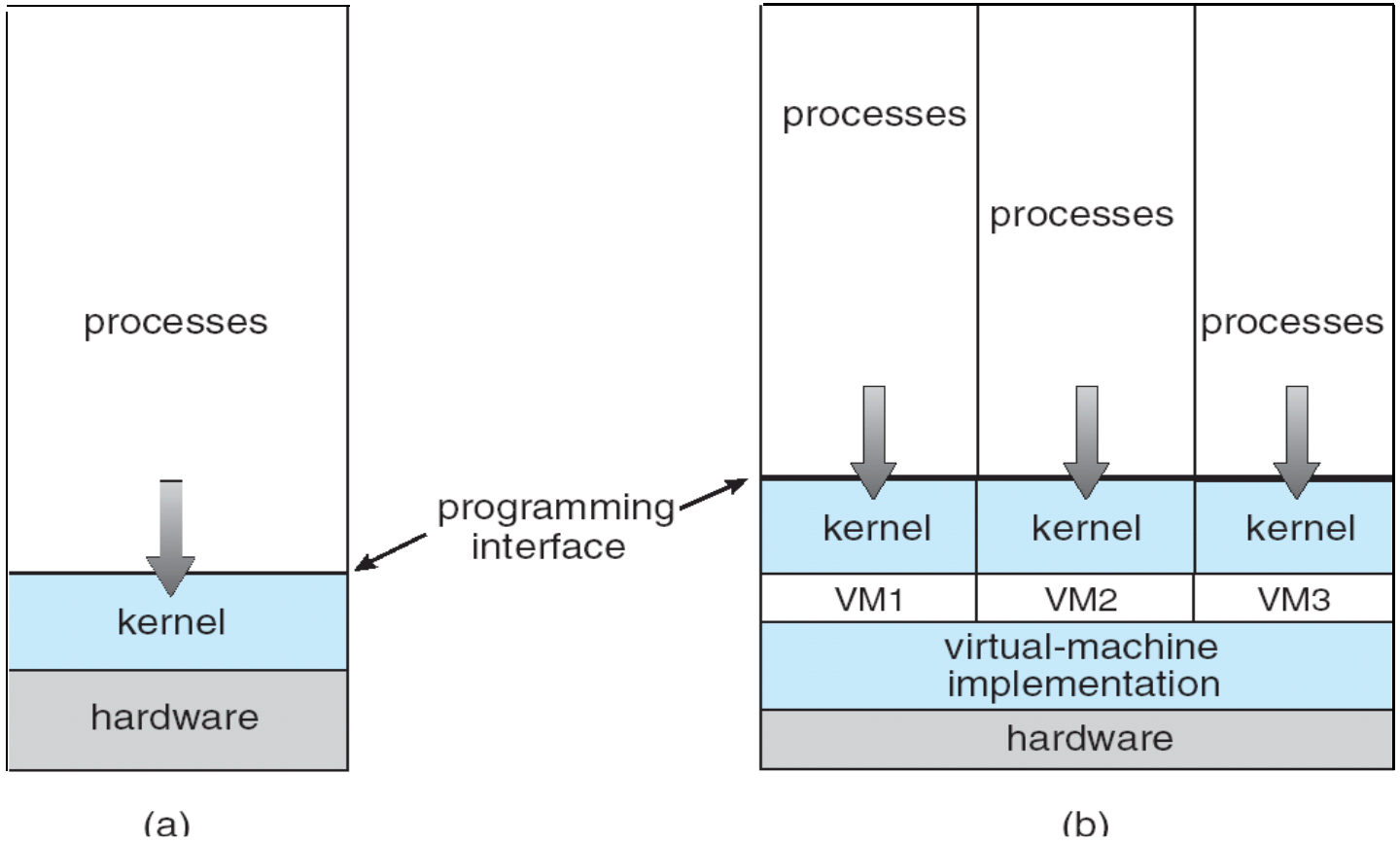
- Allows **operating systems** to run **applications** within **other OSes**
  - Vast and growing industry
- **Emulation** used when source **CPU type** different from **target type** (e.g. **PowerPC** to Intel **x86**)
  - Generally **slowest** method
  - When **computer language** not compiled to **native code** – **Interpretation**
- **Virtualization** – running **guest OSes** natively compiled for **CPU**
  - Consider **VMware** running WinXP **guests**, each running applications, all on native Win-7 **host** OS
  - **VMM** provides virtualization services

Computing Environments - Virtualization

- Laptops or desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM (virtual machine monitor) can run natively, and they are the host OS
  - VMware ESX and Citrix XenServer



Computing Environments - Virtualization



(a) **Non-virtual** machine

(b) **virtual** machine

Computing Environments – Cloud Computing

- Delivers **computing**, **storage**, even **apps** as a **service** across a **network**
- **Logical extension** of **virtualization** as based on **virtualization**
  - **Amazon EC2** has **thousands of servers**, **millions of VMs**, PBs of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - **Software as a Service** (**SaaS**) – one or more **applications** available via the Internet (e.g. word processor)
  - **Platform as a Service** (**PaaS**) – **software stack** ready for application use via the Internet (e.g. a **database server**)
  - **Infrastructure as a Service** (**IaaS**) – **servers or storage** available over Internet (e.g. storage available for backup use)

Open-Source Operating Systems

- **Operating systems** made available in source-code format rather than just binary closed-source (早期, 1950, 開放原碼盛行)
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "**copyleft**" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
  - **Linus Torvlds** (a student in Finland), released a UNIX-like kernel in **1991**.
- Can use **VMM** like ***VMware Player*** (Free on Windows), ***Virtualbox*** (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run **guest operating systems** for exploration