고려대학교
KOREA UNIVERSITY

KU-The Future

Operating Systems (10th Ed., by A. Silberschatz)

Chapter 1 Introduction

Heonchang Yu

Distributed and Cloud Computing Lab.

Contents

- 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
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

1. 운영체제라는 것은 하드웨어 디바이스를 기반으로 운영체제의 지원을 받아 애플리케이션을 실행시키는 것을 의미함

2. OS의 목표는 유저가 사용하는 프로그래밍을 좀 더 쉽고, 편리하고 효율적으로 지원하는 것

What Operating Systems Do?

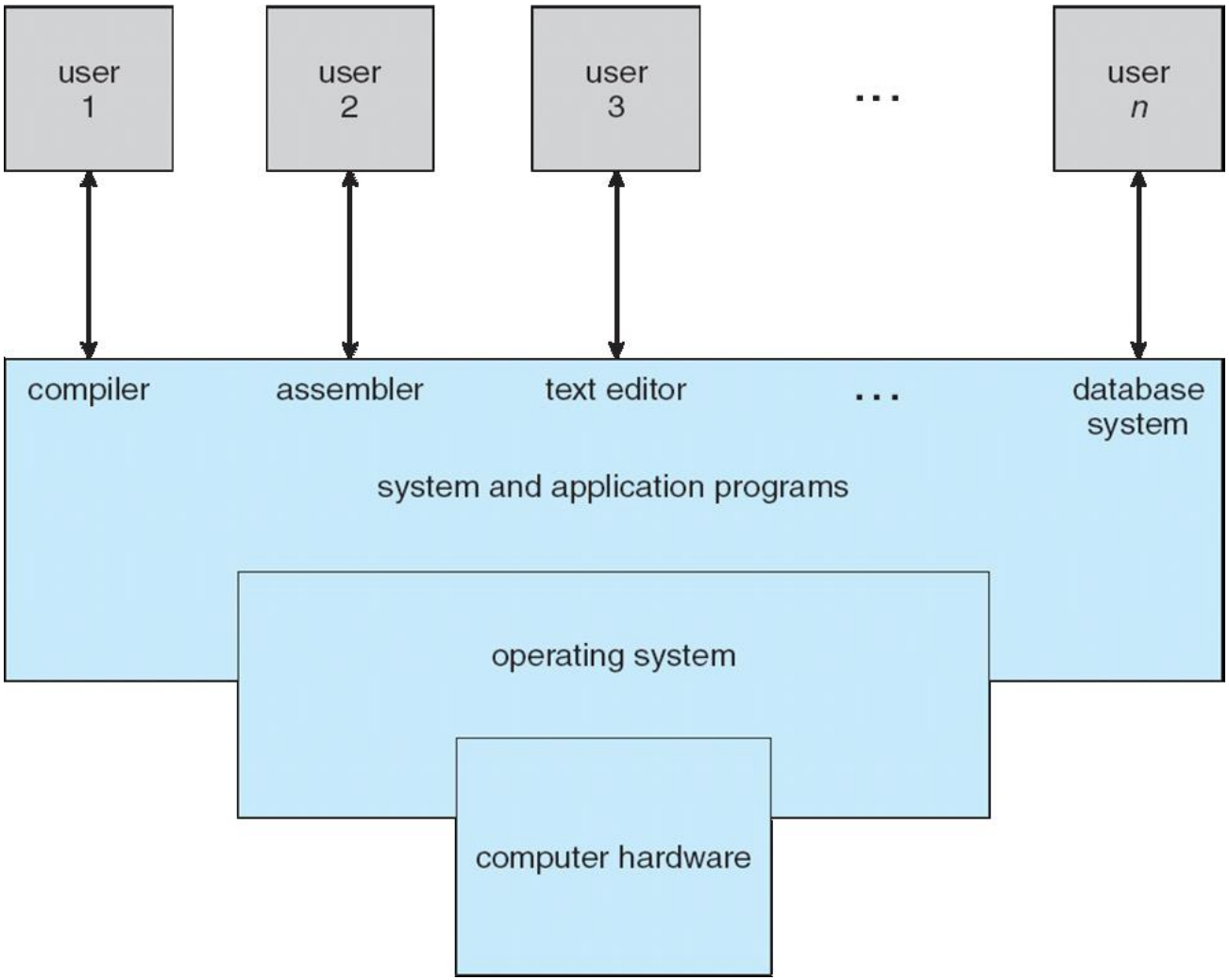
- A program that acts as an intermediary between the computer user and the computer hardware.
- Operating system goals:
  - ✓ Execute user programs and make solving user problems easier.
  - ✓ Make the computer system convenient to use.
  - ✓ Use the computer hardware in an efficient manner.

What Operating Systems Do?

– 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 – define the ways in which the system resources are used to solve the computing problems of the users
  - ✓ Word processors, compilers, web browsers, database systems, video games
- Users
  - ✓ People, machines, other computers

What Operating Systems Do?



What Operating Systems Do?

- OS is a **resource allocator** 리소스 할당은 OS에서 관리하는 모든 것이기 때문
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition

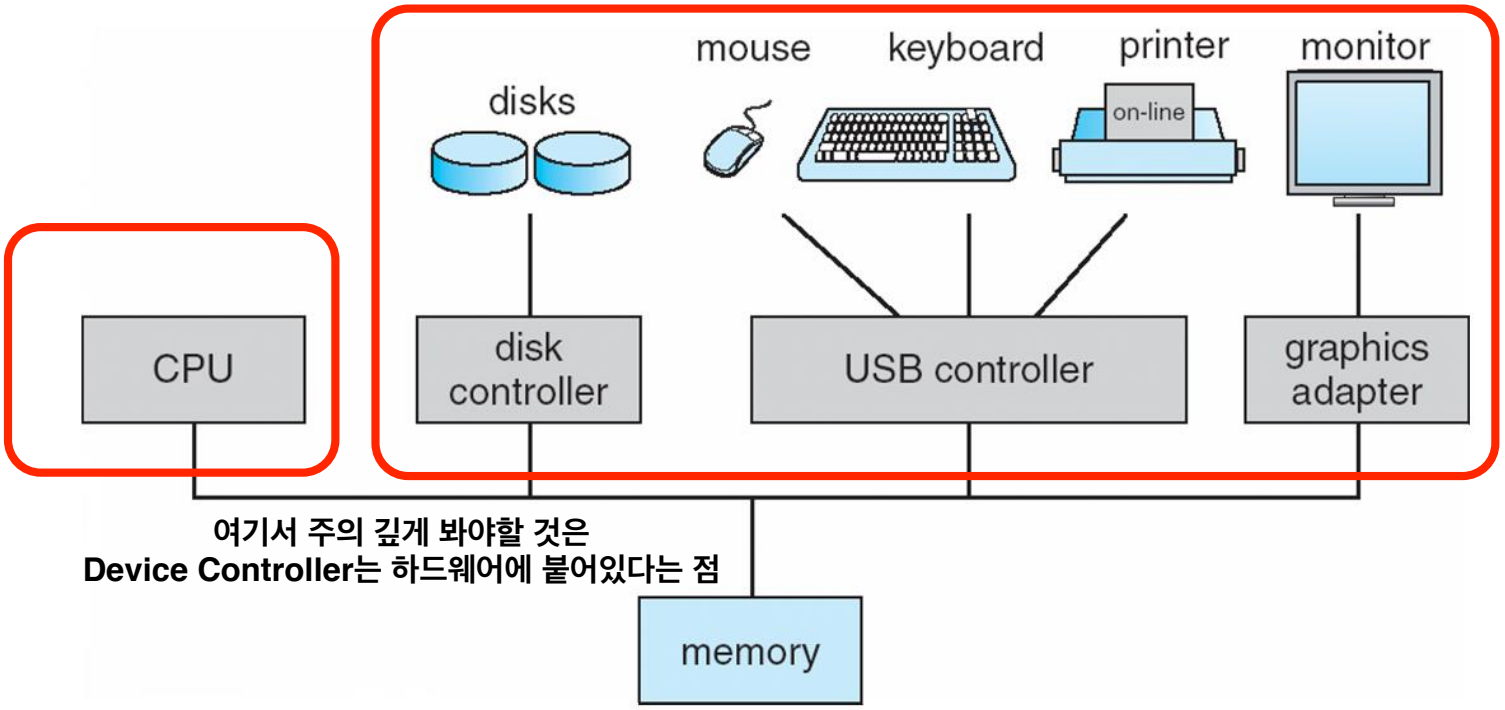
- No universally accepted definition
- It includes everything a vendor ships when you order "the operating system".
  - But varies wildly
- The one program running at all times on the computer – usually is called the **kernel.**
- Along with the kernel, there are two other types of programs: **system programs**, which are associated with the operating system but are not part of the kernel, and **application programs**, which include all programs not associated with the operation of the system.

Computer System Organization

– Computer-system operation

- 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

Device Driver는 OS에 붙어있다.



여기서 주의 깊게 봐야할 것은
Device Controller는 하드웨어에 붙어있다는 점

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as **firmware**
  - Initializates all aspects of system
  - Loads operating system kernel and starts execution

부트스트랩 프로그램은 컴퓨터를 구동시키는 프로그램이다.
마더보드 내에 **ROM**이라는 것이 있고, 펌웨어 형태로 저장되어 있다.

일반 유저가 접근할 수는 없다. 일반 유저가 물리적 업데이트 불가능
부트스트랩은 실행을 주도하는 부팅 프로그램

Computer-System Operation

- I/O devices and the CPU can execute concurrently. 동시에 수행한다 (서로 독립적)
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

CPU와 I/O 장치는 서로 독립적인 역할을 수행한다. - 상호 의존적이지 않다 (None Dependency)
CPU가 직접 로컬 버퍼를 연결하는 것이 아닌, 메모리를 통해 읽고 보내는 형태
(메인 메모리로는 incoming, 로컬 버퍼로는 outcoming)

Common Functions of Interrupts

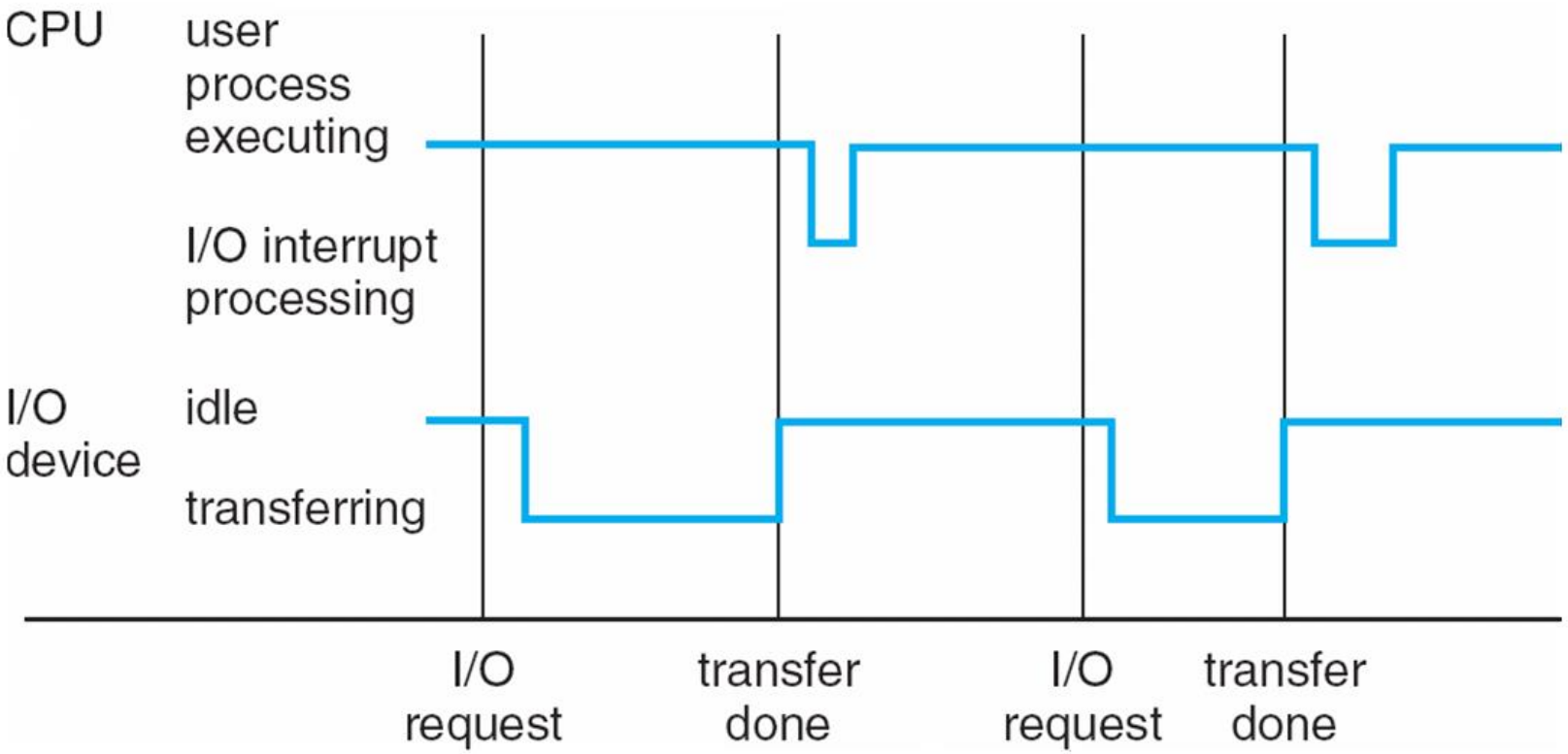
fixed location : interrupt를 시작하되는 지점. ISR 지칭

- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. (the starting address where the service routine for the interrupt is located)
- On completion, the CPU resumes the interrupted computation.
- Interrupt transfers control to the interrupt service routine.
- Interrupt architecture must save the address of the interrupted instruction.

돌아올 장소

- After the interrupt is serviced, the saved return address is loaded into the program counter, and the interrupted computation resumes as though the interrupt had not occurred.
- A *trap* or *exception* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.

Interrupt Timeline



Interrupt Timeline

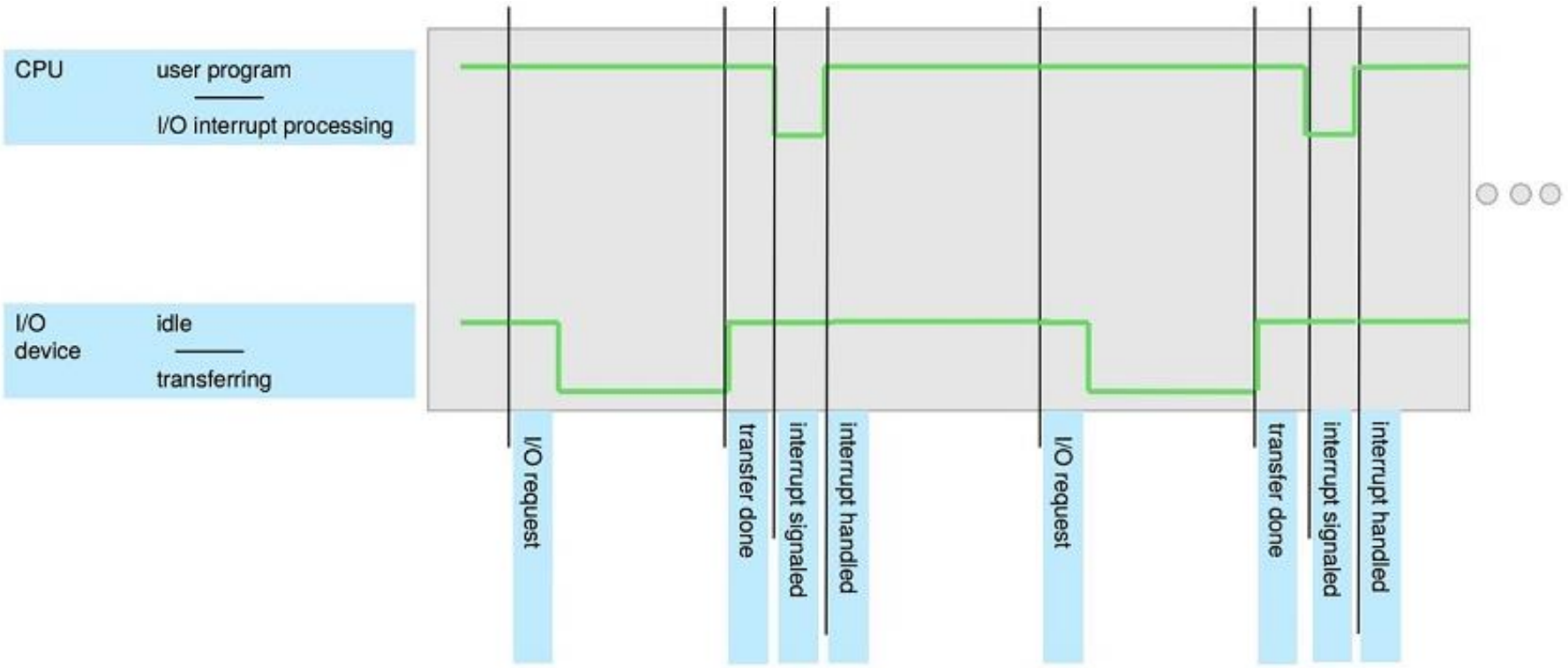


Figure 1.3 Interrupt timeline for a single program doing output.

Interrupt-driven I/O cycle

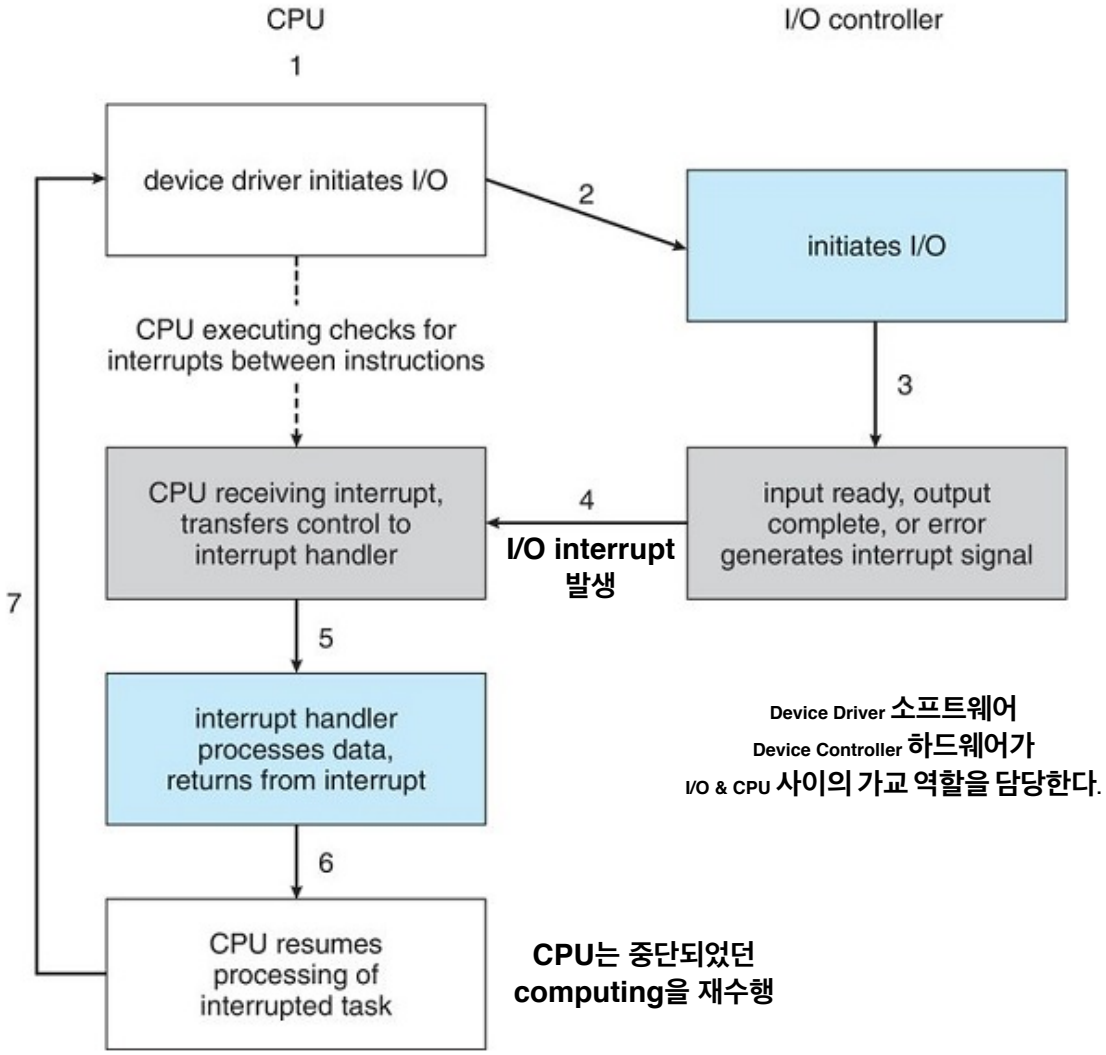


Figure 1.4 Interrupt-driven I/O cycle.

Storage Structure

단순하게 정리된 용어들.
뒤에서 다시 다룰 예정

- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Hard disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer.
- Solid-state disks – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Definitions and Notation Review

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
A **kilobyte**, or **KB**, is 1,024 bytes
a **megabyte**, or **MB**, is $1{,}024^2$ bytes
a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
a **terabyte**, or **TB**, is $1{,}024^4$ bytes
a **petabyte**, or **PB**, is $1{,}024^5$ bytes

Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Storage-Device Hierarchy

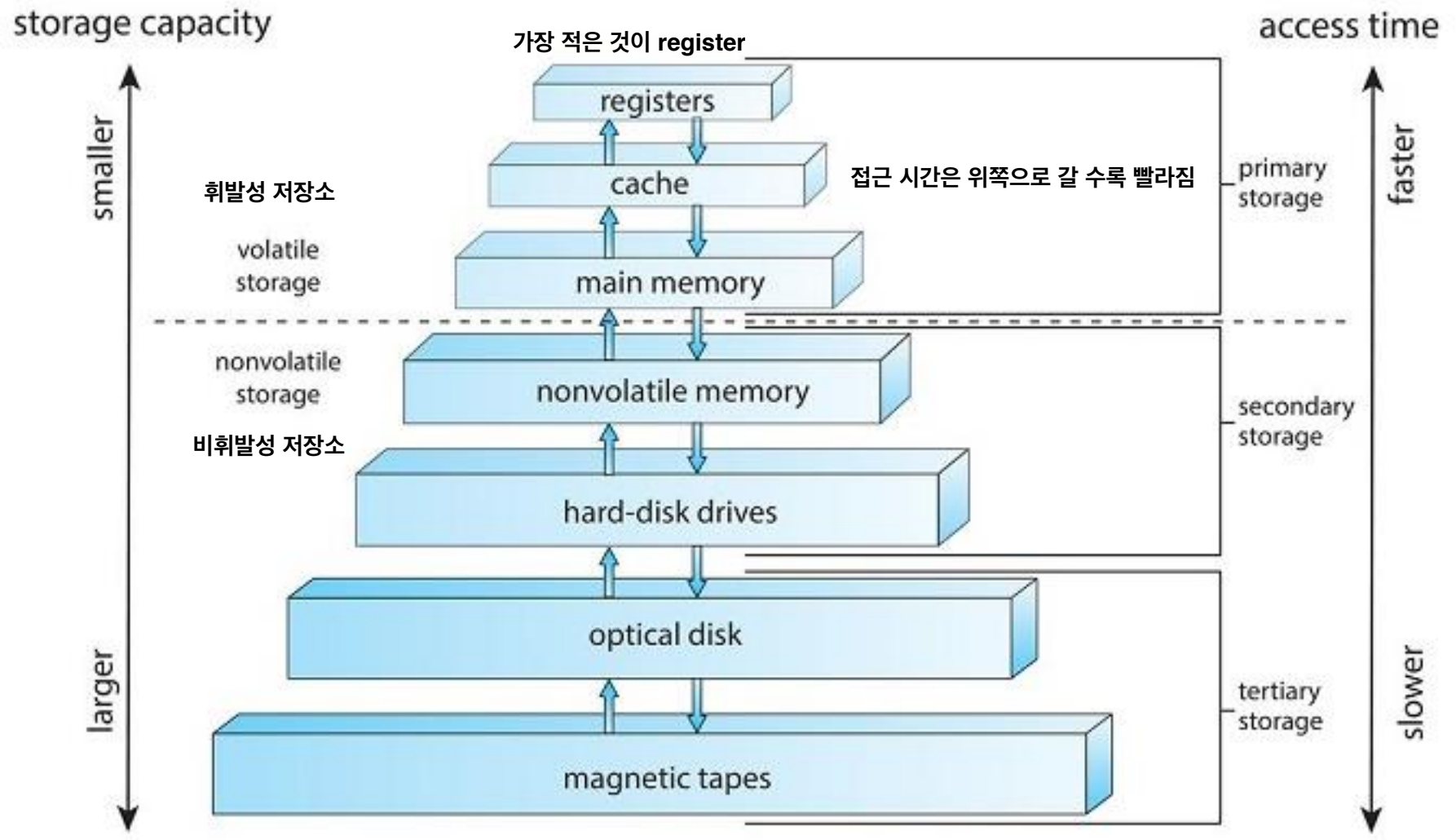


Figure 1.6 Storage-device hierarchy.

I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus. Device Controller -> Hardware
  - Each device controller is in charge of a specific type of device.
  - A device controller maintains some local buffer storage and a set of special-purpose registers. Device Driver -> OS Software
- Operating systems have a device driver for each device controller.
  - This device driver understands the device controller and presents a uniform interface to the device to the rest of the operating system.

I/O Structure

- After I/O starts, control returns to user program only upon I/O completion.
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.
- After I/O starts, control returns to user program without waiting for I/O completion.
  - *System call* – request to the operating system to allow user to wait for I/O completion. 커널 쪽에 요청하는 Operation
  - *Device-status table* contains entry for each I/O device indicating its type, address, and state.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

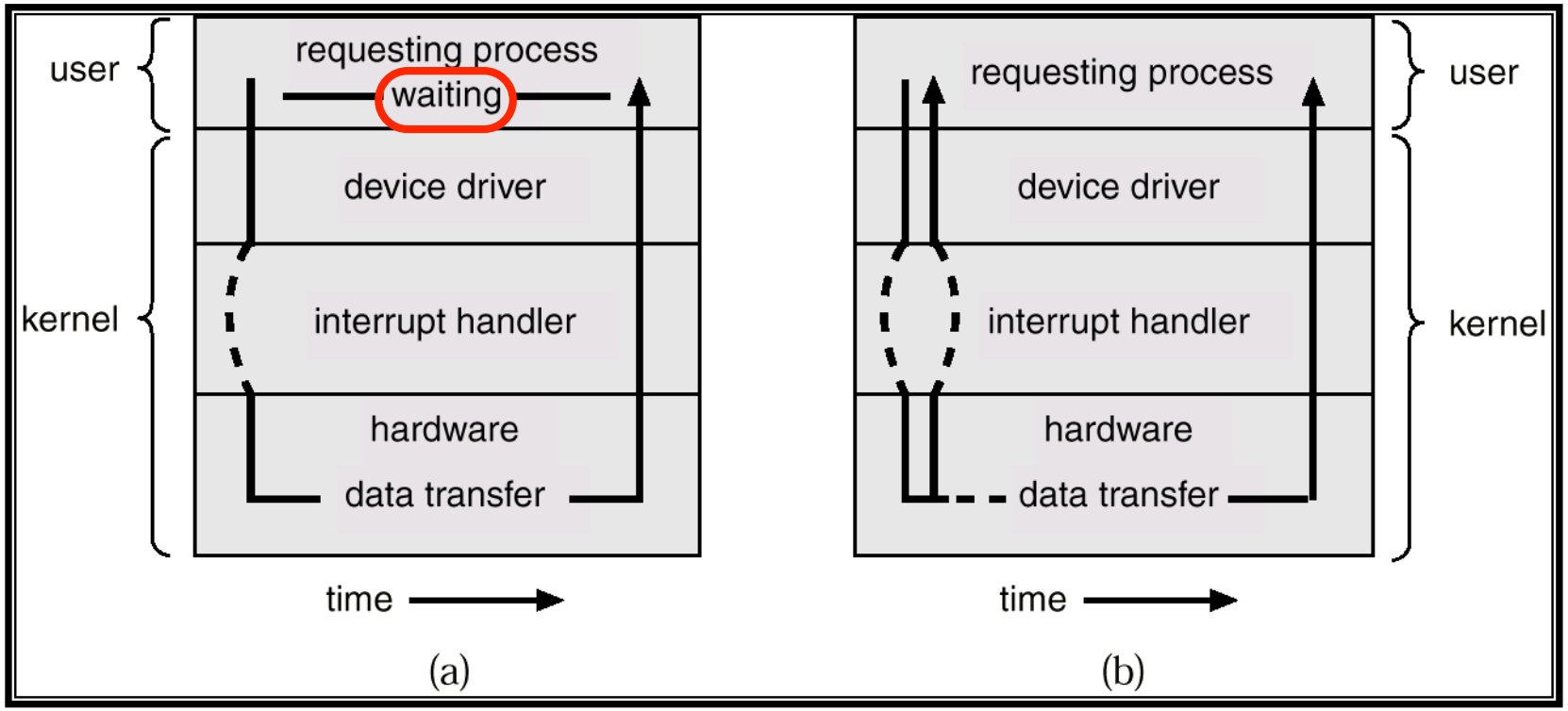
Two I/O Methods

의존성이 있다.

Synchronous

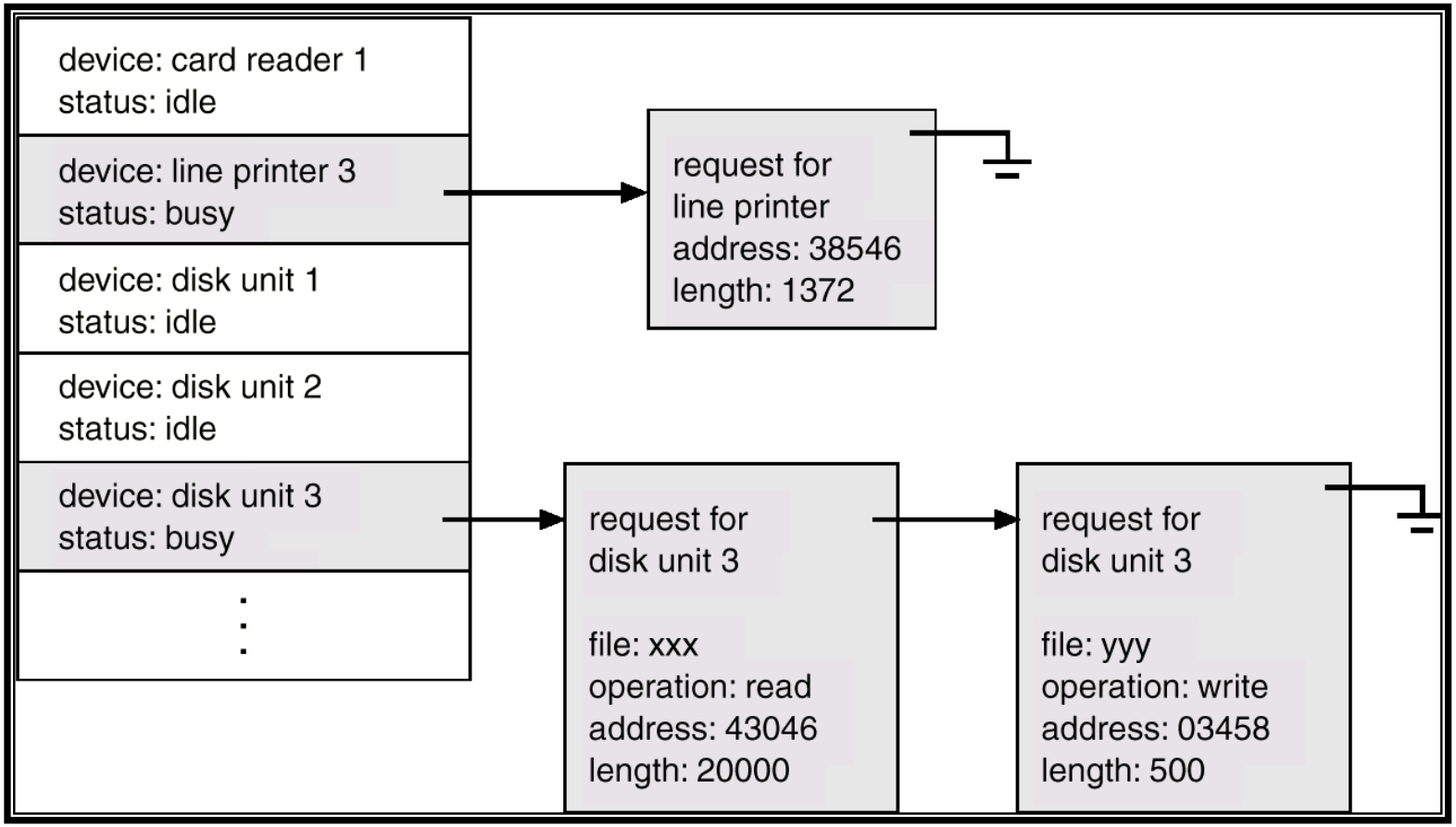
의존성 없이 동시 수행

Asynchronous



Device-Status Table

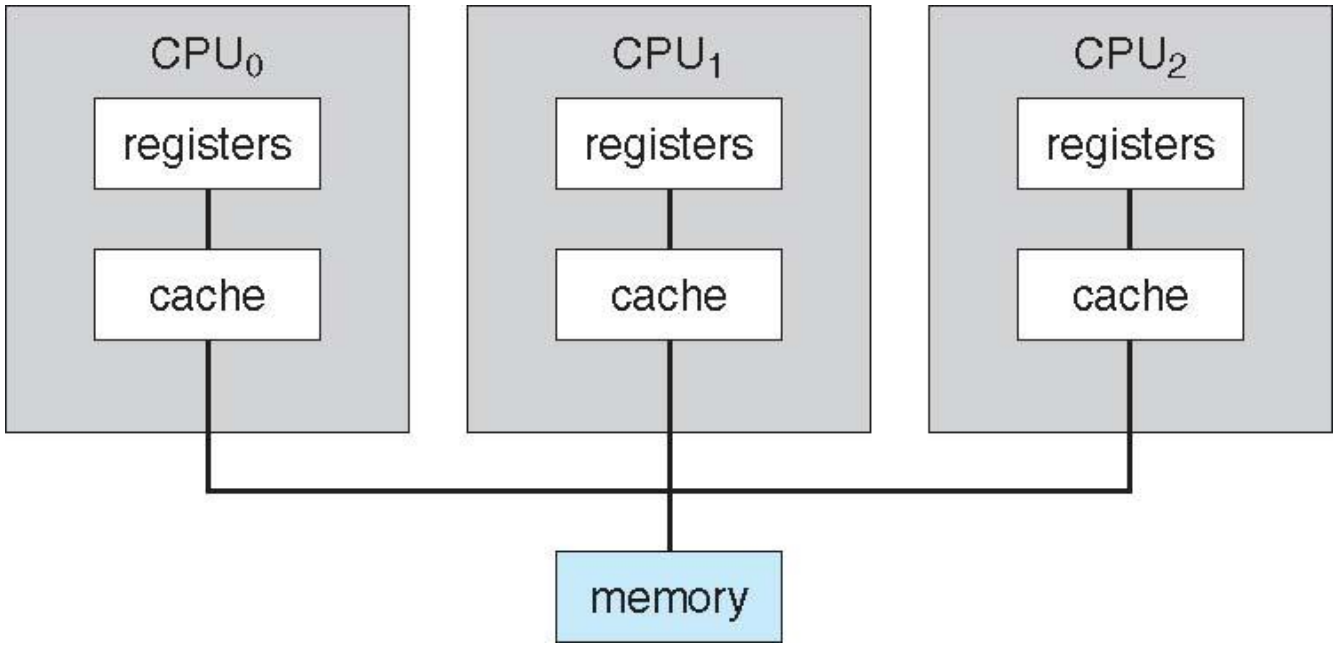
Queue처럼 이해하면 된다.



Computer-System Architecture

- Single-Processor Systems
  - ✓ On a single-processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.
- Multiprocessor Systems 단일 시스템에서 코어를 여러개 갖고 있는 형태
  - ✓ Parallel systems or tightly coupled systems
  - ✓ Advantages 반대 개념이 Loosely Coupled System
    - 처리량 증가 ❖ Increased throughput – more work done in less time
    - ❖ Economy of scale – cost less than equivalent multiple single-processor systems
    - ❖ Increased reliability – the failure of one processor will not halt the system cf.) fault tolerance fault tolerence(결함 포용) 프로세스 하나 문제 생기는 것이 시스템 전체에 문제를 만들지 않는다. —> 대체가 가능하기 때문 redunduncy
  - ✓ Asymmetric multiprocessing
    - ❖ A master processor controls the system Master - Slave 관계
    - ❖ The master processor schedules and allocates work to the slave processors
  - ✓ Symmetric multiprocessing (SMP) 프로세스들 간의 관계가 동등
    - ❖ All processors are peers; no master-slave relationship

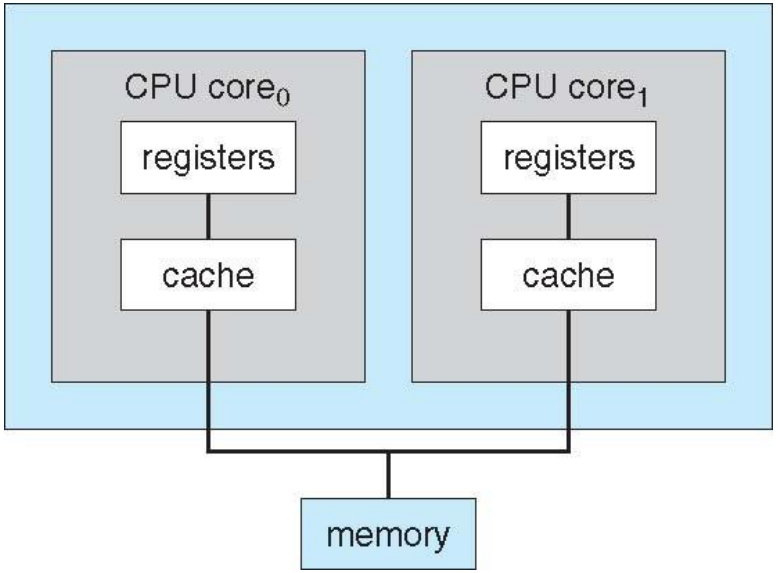
Symmetric Multiprocessing Architecture



A Dual-Core Design

Uniform Memory Access / None Uniform Memory Access

- **UMA** and **NUMA** architecture variations 다른 코어에 있는 메모리의 접근을 허용하는 게 NUMA
- Multi-chip and **multicore**
- Systems containing all chips vs. **blade servers**
  - Chassis containing multiple separate systems



Loosely Coupled System

- 각 프로세스마다 독립된 메모리를 각지 시스템으로, 분산처리 시스템이라고도 한다.
- 둘 이상의 독립된 컴퓨터 시스템을 통신망(통신 링크)을 통하여 연결한 시스템이다.
- 각 시스템마다 독자적인 운영체제를 가지고 있다.
- 각 시스템은 독립적으로 작동할 수 있고, 필요한 경우에는 상호 통신을 할 수도 있다.
- 프로세스 간의 통신은 메시지 전달이나 원격 프로시저 호출을 통해서 이루어 진다.
- 각 시스템마다 독자적인 운영이 가능하므로 프로세서 간이 결합력이 약하다.

Clustered Systems

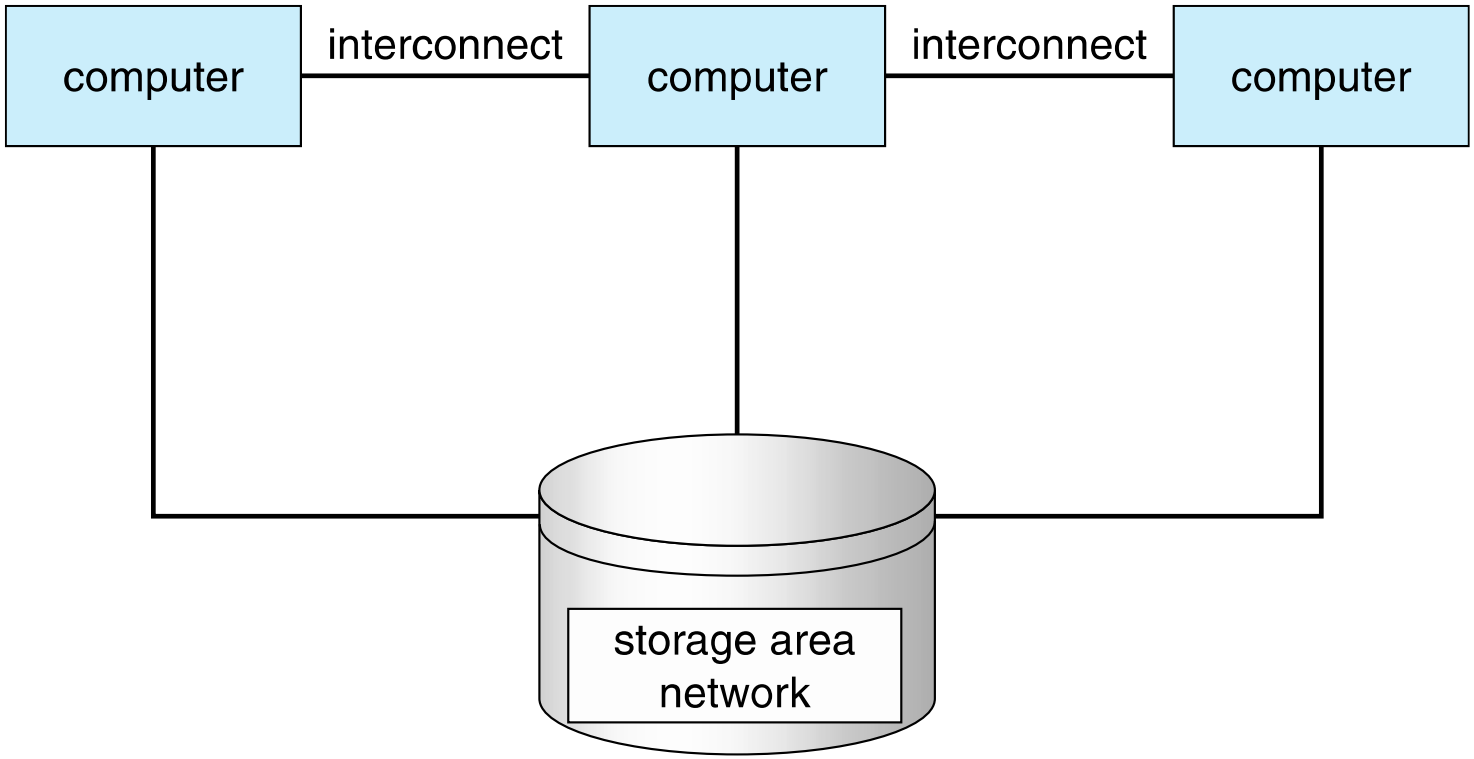
분산 컴퓨팅 시스템은 일반적으로 Loosely Coupled System임

멀티 프로세스는 Tightly Coupled System

- Like multiprocessor systems, but multiple systems working together 멀티 프로세서처럼 시스템도 멀티로 운영이 가능하다.
  - Usually sharing storage via a **storage-area network (SAN)**
  - 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)** 슈퍼 컴퓨터가 대표적인 예시다. 고성능 컴퓨팅
    - ✓ Applications must be written to use **parallelization**
  - Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

어떤 자원을 sharing하는 것은 굉장히 좋지만, 해결할 것이 많은데 대표적인 것이 Synchronization임

Clustered Systems

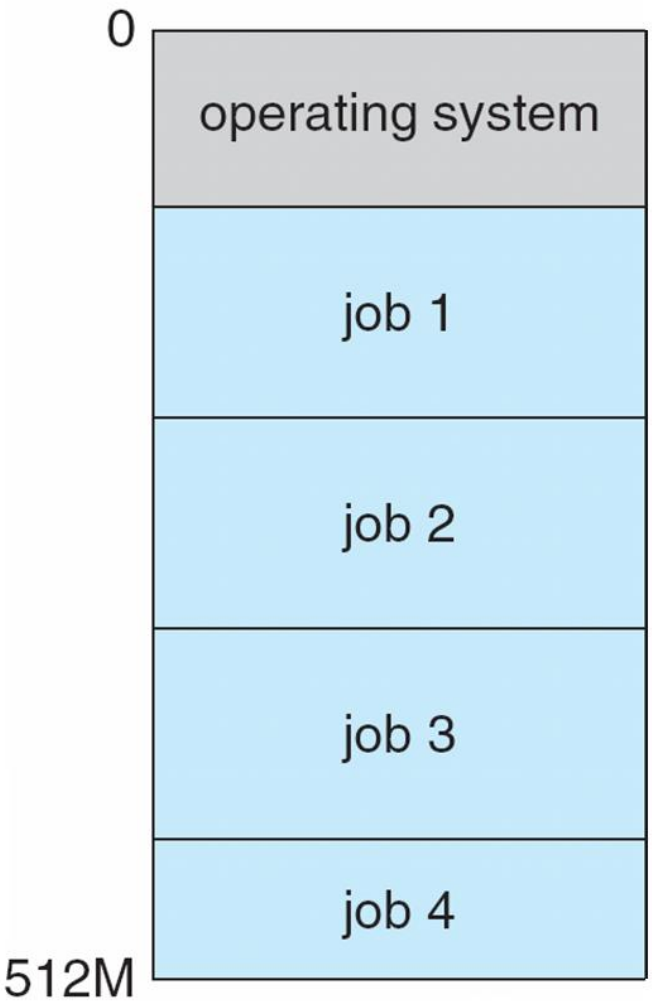


Operating-System Structure

- **Multiprogramming** needed for efficiency 상반 되는 개념은 single programming
  - ✓ Single user cannot keep CPU and I/O devices busy at all times
  - ✓ Multiprogramming organizes jobs (code and data) so CPU always has one to execute CPU가 idle상황에 들어가면 task를 2개 이상 상주시켜 또다른 작업을 할당하는 방식. 이러한 방식을 스케줄링이라 함 I/O와 연결지어 생각하면 안된다!
  - ✓ A subset of total jobs in system is kept in memory
  - ✓ One job selected and run via **job scheduling**
  - ✓ When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)** is logical extension in which 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**
  - ✓ 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
  - ✓ **Virtual memory** allows execution of processes not completely in memory

작업을 선정해서 올려주는 게 job scheduling

Memory Layout for Multiprogrammed System



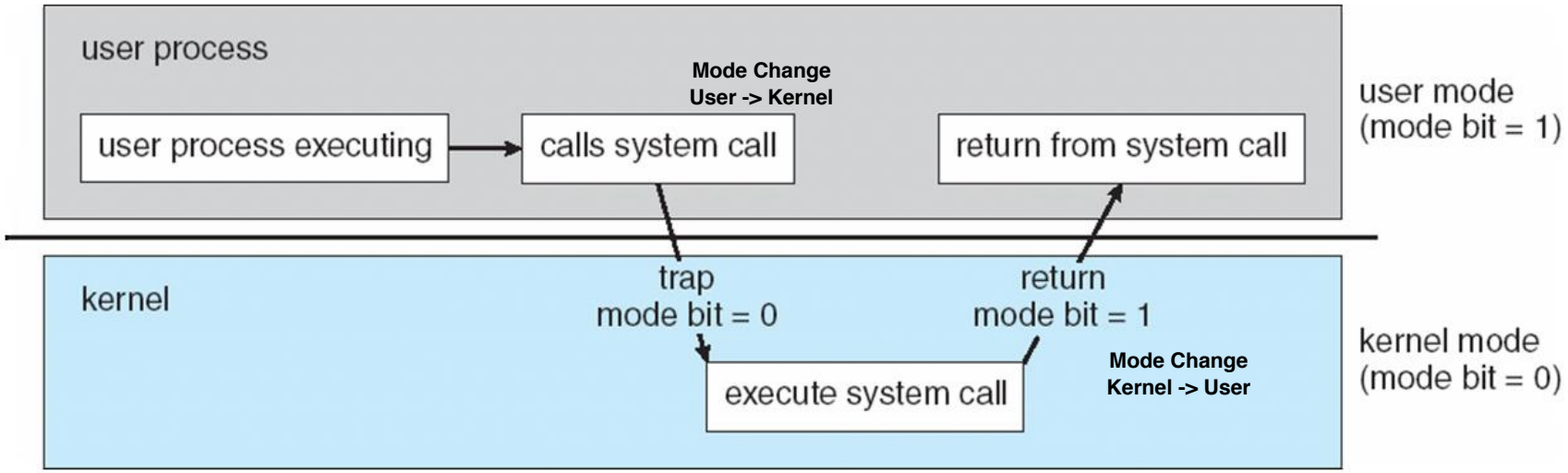
Operating-System Operations

- Modern operating systems are Interrupt driven.
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components OS 영역과 유저 영역을 넘나드는 것
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - ✓ Provides ability to distinguish when system is running user code or kernel code
      커널 영역이 privileged되었다고 표현하는 것이다.
    - ✓ Some instructions designated as **privileged**, only executable in kernel mode
    - ✓ System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set to interrupt the computer after a specified period
  - Operating system decrements counter
  - When the counter reaches 0, an interrupt occurs.
  - If the time interrupts, control transfers automatically to OS, which may treat the interrupt as a fatal error or may give the program more time.

무한루프를 방지하기 위해 카운터 값이 0이 되면 인터럽트 발생



Process Management

프로세스는 메모리에 저장되는 것이고, 프로그램은 하드웨어에 저장되는 것
하드웨어에 있는 것이 메모리에 로딩되면 프로세스이다.

- A process is a program in execution. It is a unit of work within the system. 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 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

교착 상태

동시에 뭔가 이뤄진다면, 서로 다른 프로세스들이 리소스를 사용하려고 하는 일이 발생할 것이다.

이를 DB에서는 동시성 제어라고 표현한다.

프로세스들간의 통신 —> Inter Process Communication

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
  - ✓ Improving both the utilization of the CPU and the speed of the computer's response to its 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

Memory Management —> Paging

Storage Management

운영체제가 제공하는 중요한 기능 중 하나!

- OS provides uniform, logical view of information storage
  - ✓ Abstracts physical properties to logical storage unit - **file**
  - ✓ Each medium is controlled by device (i.e., disk drive, tape drive)
    - ❖ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - ✓ 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 directories
    - ❖ Mapping files onto secondary storage
    - ❖ Backup files onto stable (non-volatile) storage media

Mass-Storage Management

조직에서 대용량의 서버를 운영할 때

- 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.
- Proper management is of central importance
- 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 storage includes optical storage, magnetic tape
  - ✓ Still must be managed
  - ✓ Varies between WORM (write-once, read-many-times) and RW (read-write)

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied 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 smaller than storage being cached
  - Cache management is an important design problem
  - Cache size and replacement policy

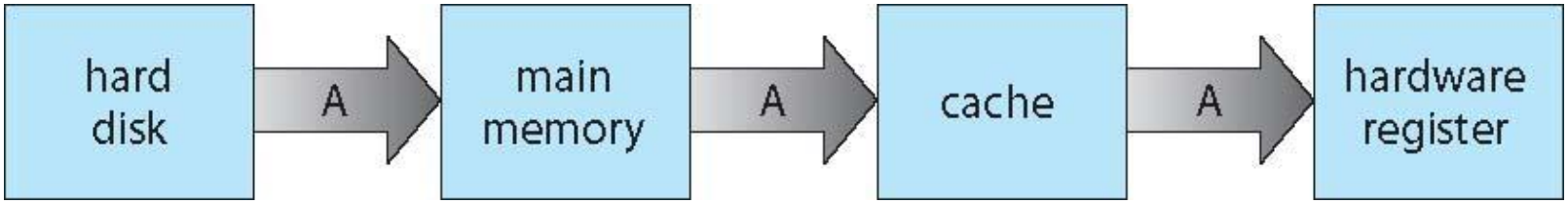
Performance of Various Levels of Storage

– Movement between levels of storage hierarchy can be explicit or implicit.

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, not matter where it is stored in the storage hierarchy



서로다른 코어에서 서로 다른 로컬 캐시를 가져가 다른 값을 유지할 수 있는 경우가 있다.

- 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
  - Various replicas may be accessed and updated concurrently.

I/O Systems

- One of the purposes of OS is to hide the peculiarities of specific hardware devices from the user.
- I/O subsystem consists of several components:
  - A memory-management component that includes buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - A general device-driver interface
  - Drivers for specific hardware devices

앞서 배웠던 Interrupt가 여기서 다루는 내용이다.

Protection and Security

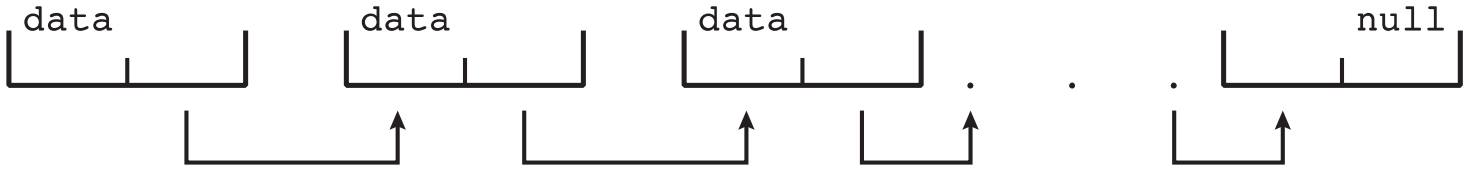
초기엔 보안 문제가 중요하지 않았다

- **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
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (g**roup ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights (to gain extra permissions for an activity)

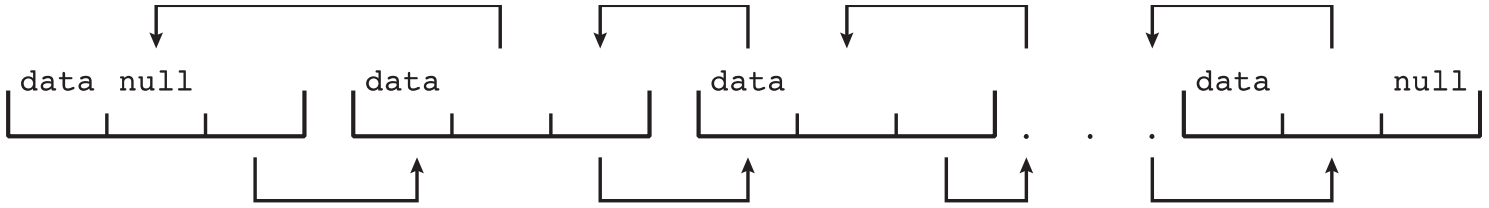
Kernel Data Structures

배열 구조는 static하기 때문에 사용하기 어려움

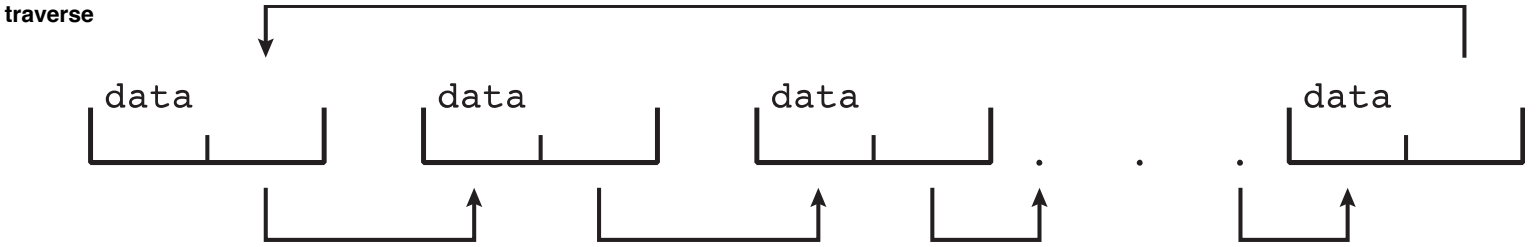
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***



- ***Circular linked list***



Kernel Data Structures

- **Tree**
  - A data structure that can be used to represent data hierarchically
- **Binary tree**
  - A parent may have at most two children
- **Binary search tree** 탐색 속도를 높이기 위함임
  left.child <= right.child
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*

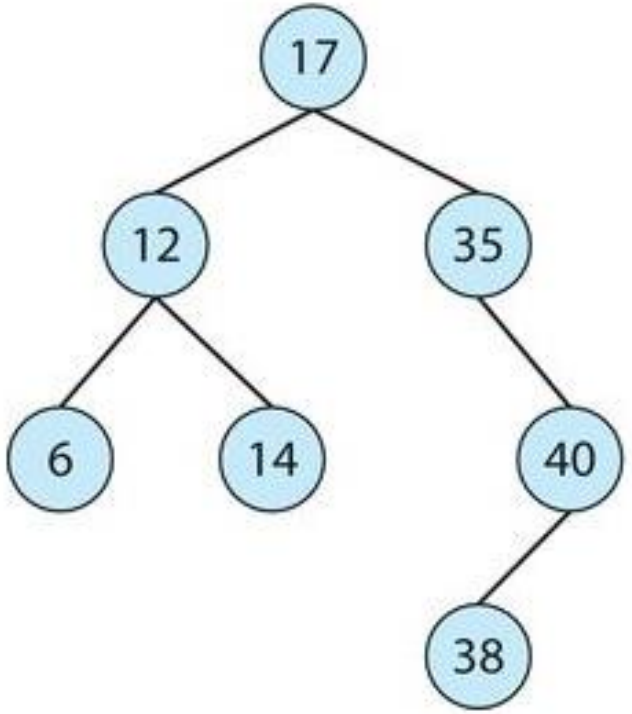
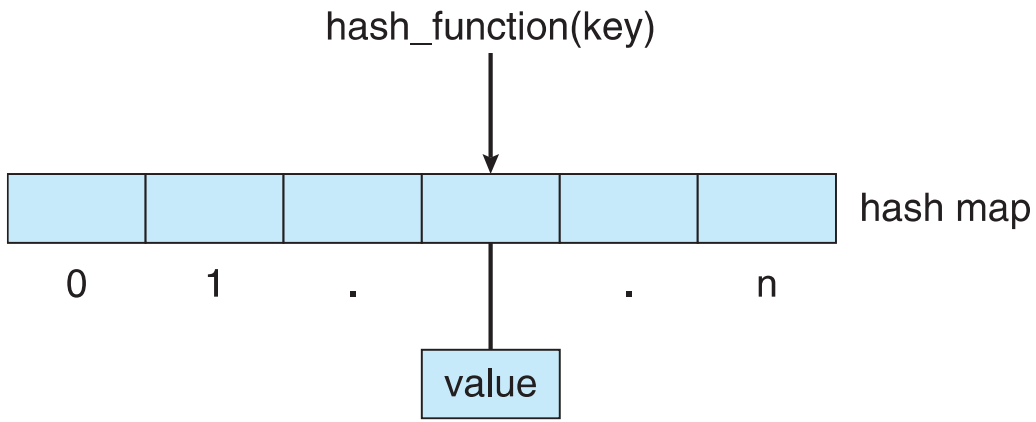


Figure 1.20 Binary search tree.

Kernel Data Structures

- **Hash function** can create a **hash map**

주어진 목록에서 접근 속도를 높이기 위함임



Paging에서 다시 언급 될 내용

- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Computing Environments

빠르게 생략된 페이지임

- Traditional Computing
  - 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** (**thin clients**) 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

빠르게 생략된 페이지임

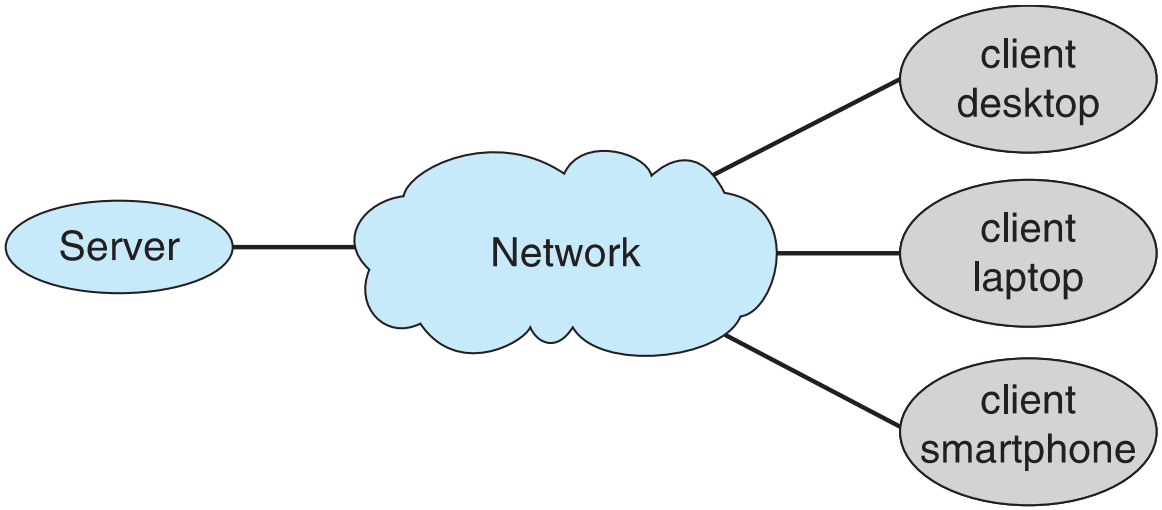
- Mobile Computing
  - Handheld smartphones, tablets, etc
  - What is the functional difference between them and a “traditional” laptop?
  - Extra feature – more OS features (GPS, gyroscope)
  - Allows new types of apps like ***augmented reality***
  - Use IEEE 802.11 wireless, or cellular data networks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

Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide the users with access to the various resources that the system maintains.
- Distributed systems depend on networking for their functionality.
  - Local Area Network (LAN)
  - Wide Area Network (WAN)
  - Metropolitan Area Network (MAN)
  - Personal Area Network (PAN)
- Network operating system
  - An operating system that provides features such as file sharing across the network and that includes a communication scheme that allows different processes on different computers to exchange messages.

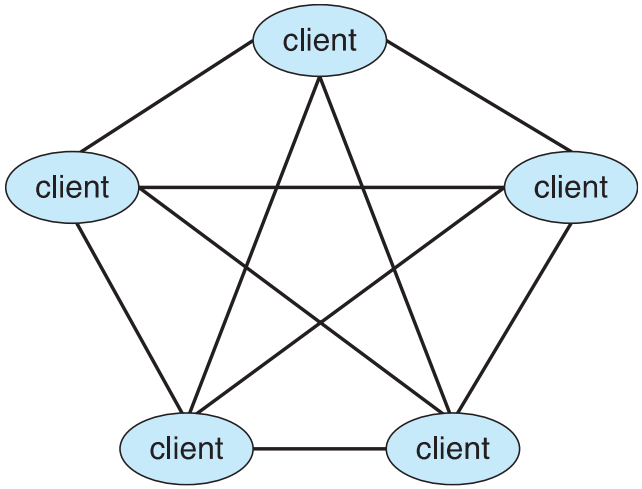
Client-Server Computing

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- 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



인터넷 프로토콜 수업과 밀접하게 연관되는 부분
분산 컴퓨팅 시스템에는 인터넷도 반드시 포함되어야한다.

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 (i.e. PowerPC to Intel x86)
  - ✓ Generally slowest method
  - ✓ When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - ✓ Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - ✓ **VMM** provides virtualization services

Computing Environments - Virtualization

- Use cases involve laptops and 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 can run natively, in which case they are also the host
  - ✓ There is no general purpose host then (VMware ESX and Citrix XenServer)

Computing Environments - Virtualization

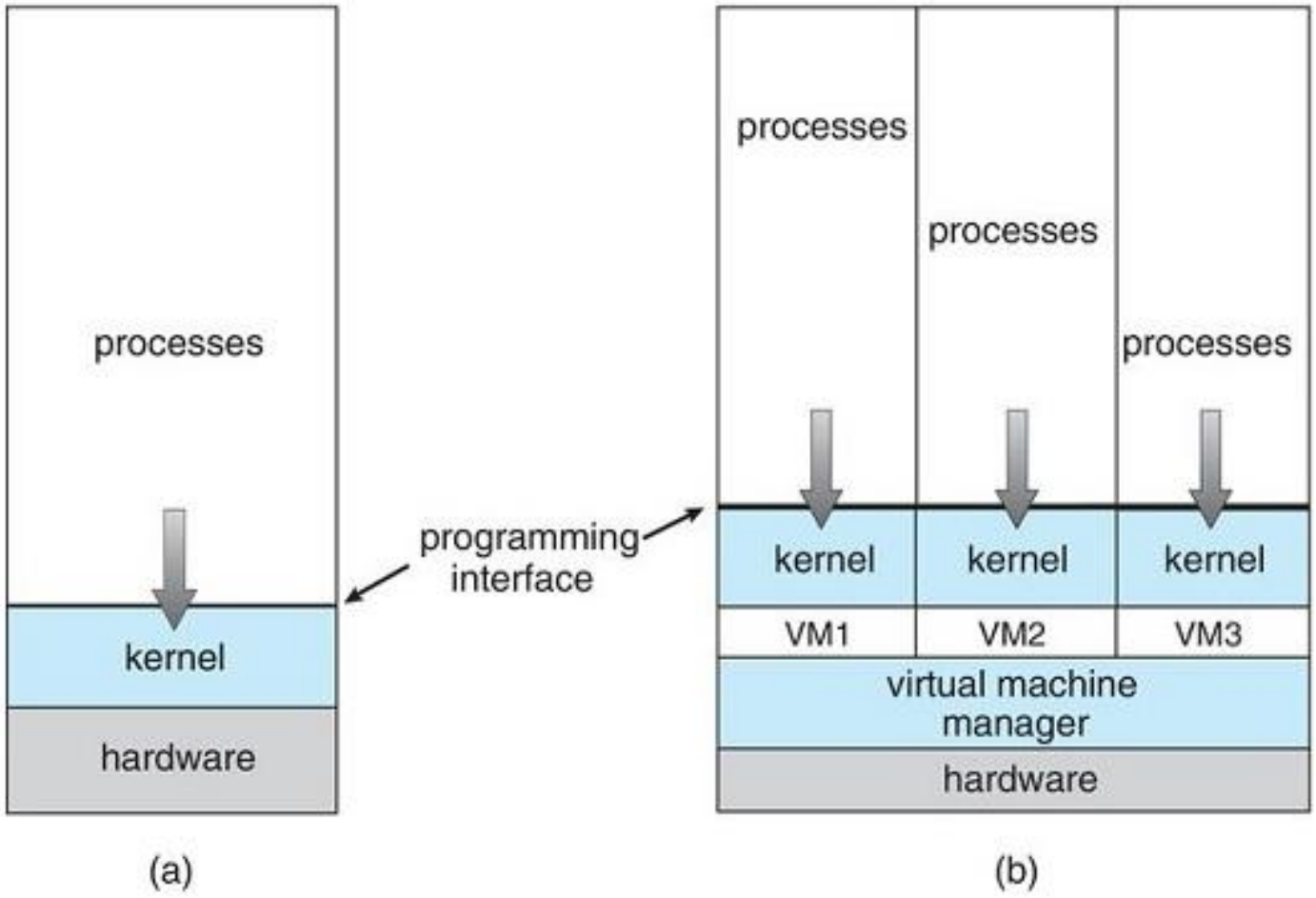


Figure 1.16 A computer running (a) a single operating system and (b) three virtual machines.

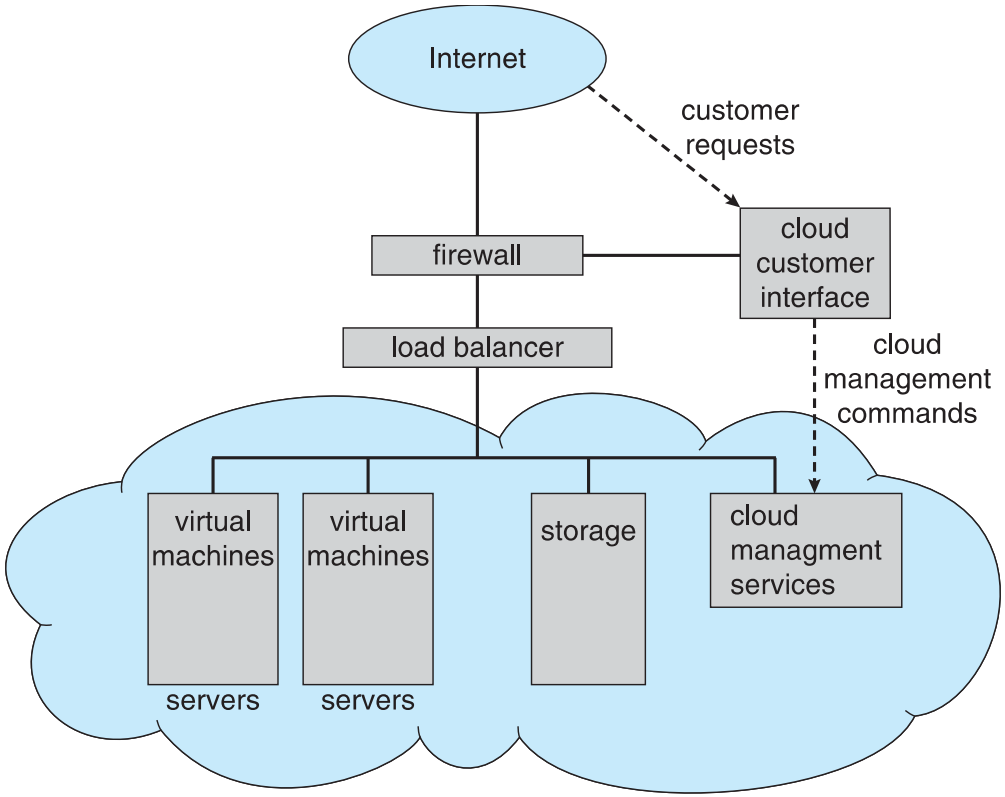
Computing Environments – Cloud Computing

클라우드 컴퓨팅이 분산 컴퓨팅을 기초로 하고 있다

- 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, petabytes 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 (i.e. word processor)
  - ✓ Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e a database server)
  - ✓ Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e. storage available for backup use)

Computing Environments – Cloud Computing

- Cloud compute environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - ✓ Internet connectivity requires security like firewalls
  - ✓ Load balancers spread traffic across multiple applications



Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - ✓ Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - ✓ Use expanding
- Many other special computing environments as well
  - ✓ Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - ✓ Processing ***must*** be done within constraint
  - ✓ Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- 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
- 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