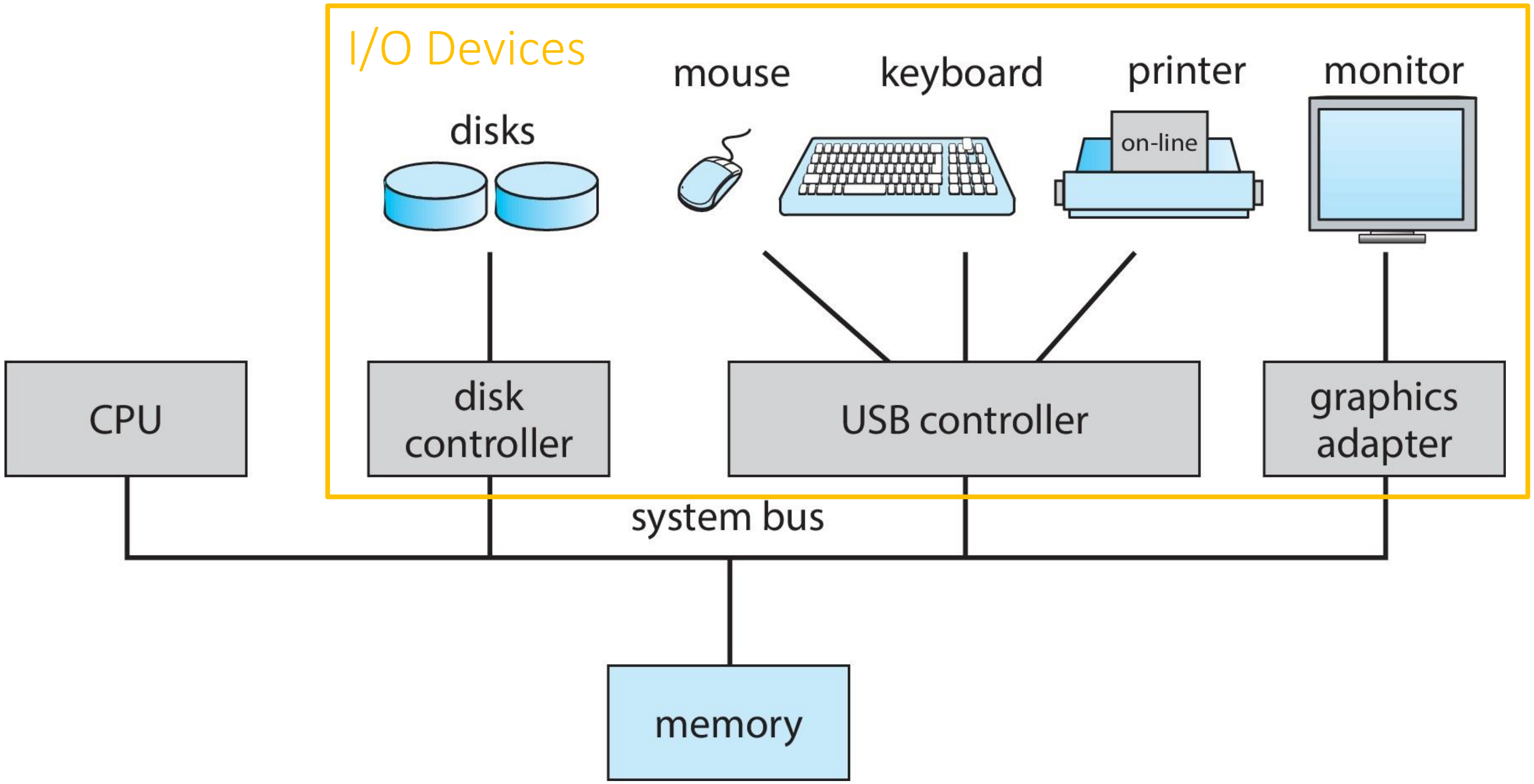
Chapter 1: Introduction

Outline

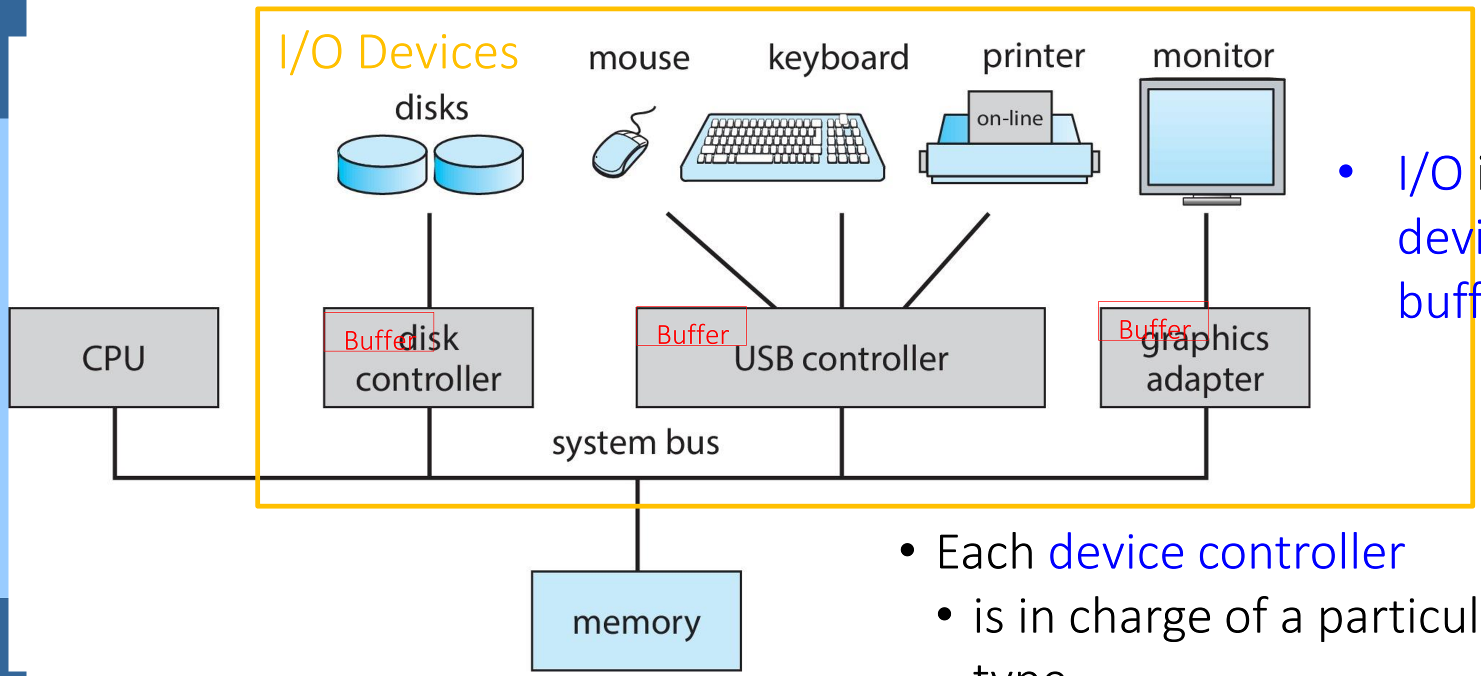
- Computer System Architecture
- Operating Systems
  - Definition
  - Goals
  - Operations
  - Types
- Virtualization
- Free and Open Operating Systems
- Main Data Structures Used in OS

Computer System Architecture



I/O Devices in Computer

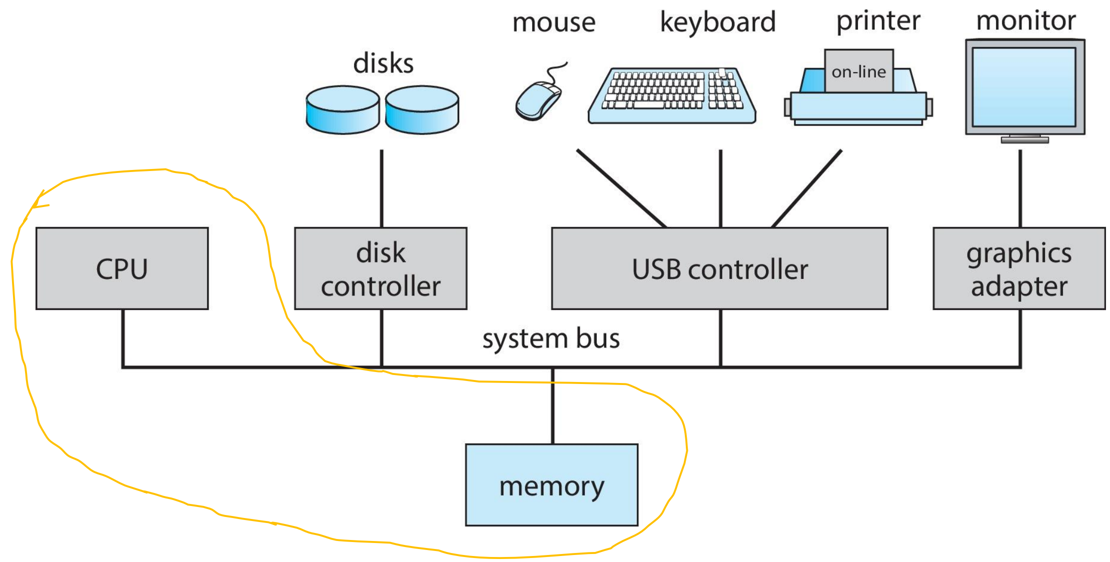
- I/O devices and the CPU can work concurrently



- I/O is between the device and local buffer of controller

- Each device controller
  - is in charge of a particular device type
  - has a local buffer (local memory)

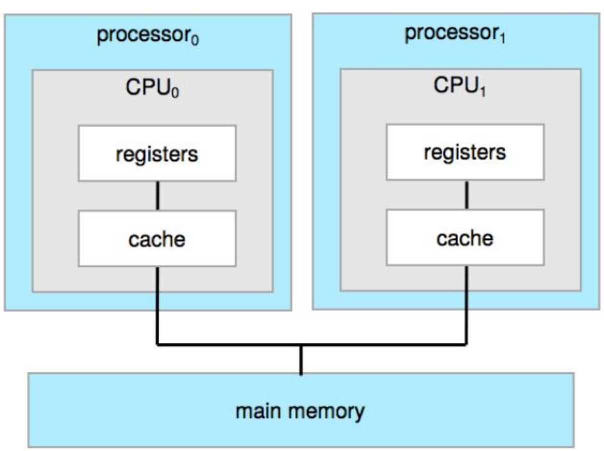
CPU and Memory in Computer



Single processor

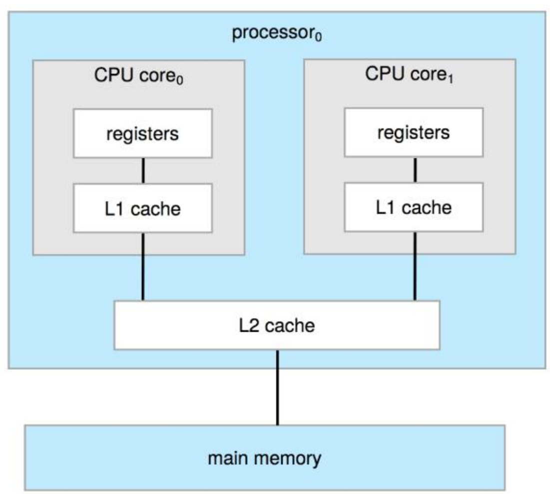
or

Multiple processors



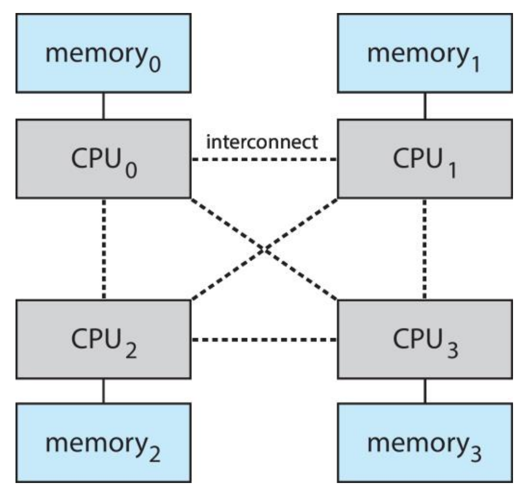
or

Dual core

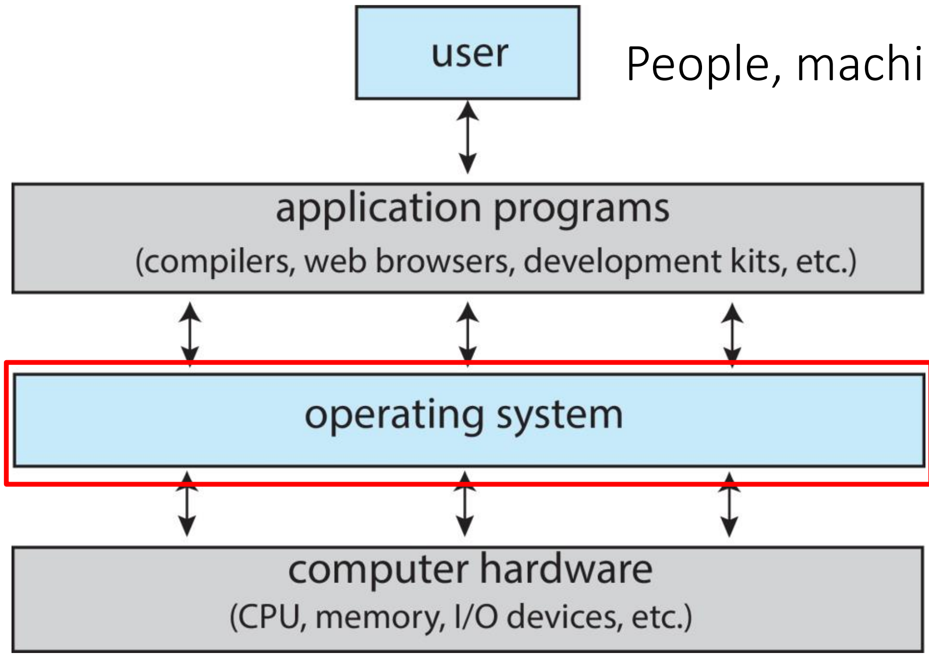


or

Non-Uniform Memory Access System (NUMA)



Operating System: Intermediary between a User and the Computer Hardware



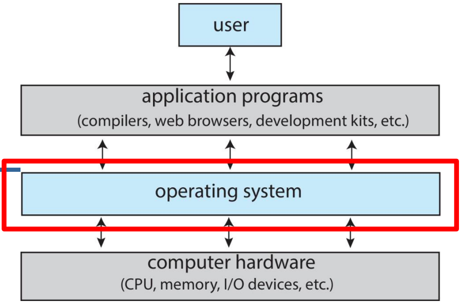
People, machines, other computers

E.g., Word processors, compilers, web browsers, database systems, video games

Controls and coordinates use of hardware among various applications and users

Provides basic computing resources

Operating Systems



An operating system

- is a program that acts as an intermediary between a user of a computer and the computer hardware
- provides interface to a user, runs a program for a user

General goals of operation systems

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

Picture source: from bing.com

Operating Systems Goals

Depends on computer/device types

- Personal computer
  - Feature: owned by individuals
  - OS goal: convenience, ease of use and good performance
- Shared computer (e.g. mainframe or minicomputer)
  - Feature: shared by multiple users
  - OS goal: keep all happy, efficient use of Hardware and managing user programs

What Operating Systems Do

Depends on computer/device types (cont.)

Mobile devices (e.g. smartphones and tablets)

- Feature: resource poor
- OS goal: optimized for usability and battery life, provide user interfaces such as touch screens, voice recognition

Embedded computers

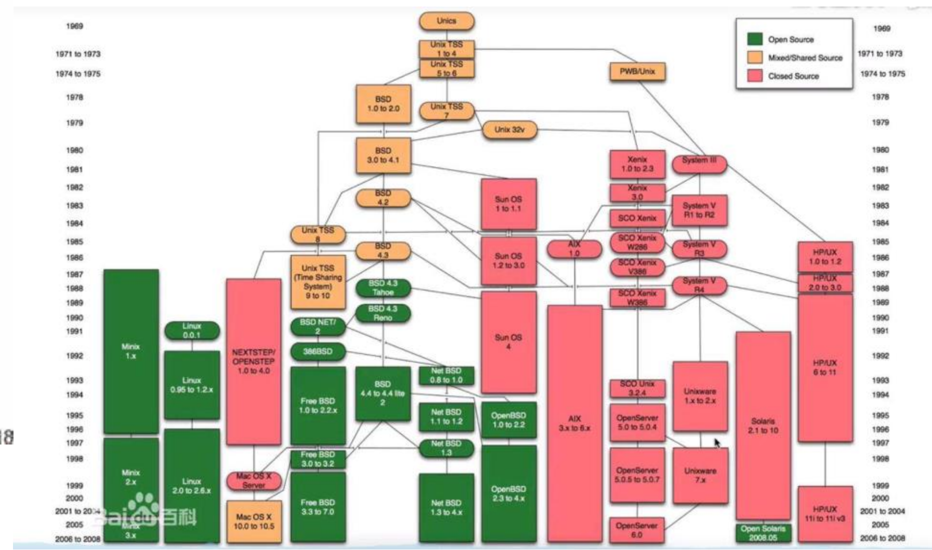
- Feature: little or no user interface
- OS goal: run program

User view: ease of use
System view: resource allocation

Operating System Examples

Term OS exists in many devices.

There is a large number of OS designs



Linux

Picture resoutce:
https://cn.bing.com/images/search?view=detailV2&ccid=m%2fuR5GOU&id=298444024B0761F4FD9628A1F3C02D0CF00DDB3B&thid=OIP.m_uR5GOUIh7Mbc2NbXowUAHaFj&mediaurl=https%3a%2f%2fimg00.deviantart.net%2fece9%2fi%2f2018%2f021%2fd%2fc%2fwindows_family_photo_by_jaceu-dc0r7j6.png&exph=450&expw=600&q=windows+family+pictures&simid=608037459554172544&FORM=IRPRST&ck=95ED26317FF82815C67EE21D7145788F&selectedIndex=0&idpp=overlayview&ajaxhist=0&ajaxserp=0https://baike.baidu.com/item/MacOS/8654551
https://baike.baidu.com/item/MacOS/8654551

Operating System Definition

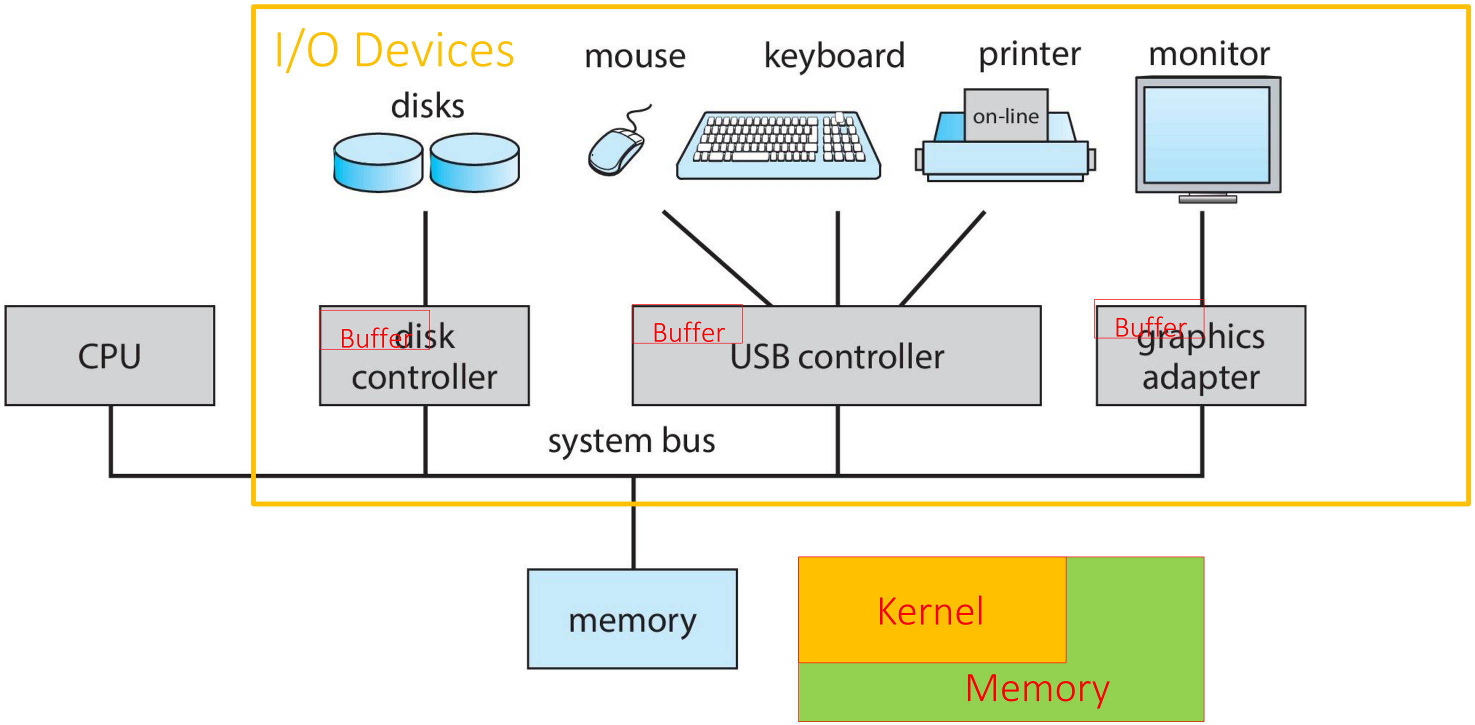
No universally accepted definition

A definition: "Everything a vendor ships when you order an operating system"

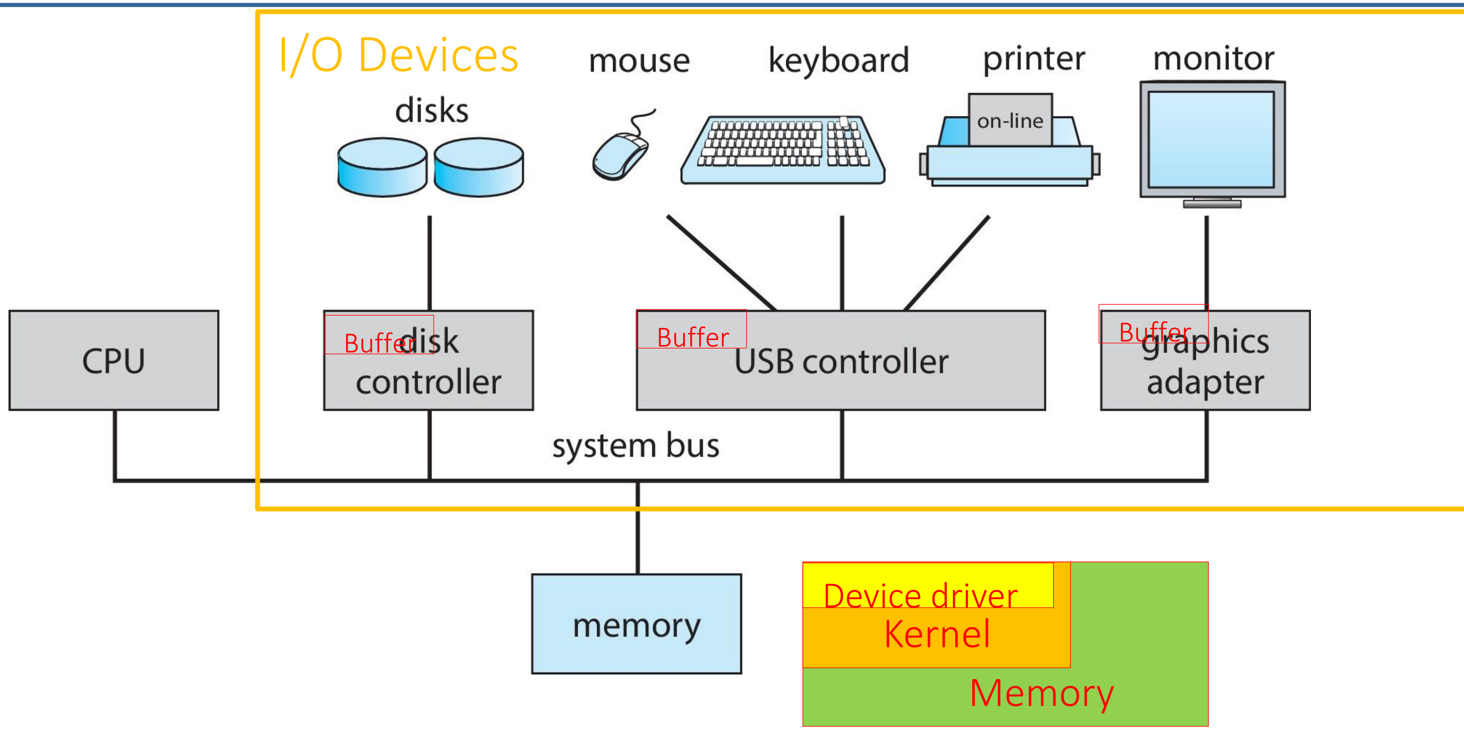
"Everything" include

- Kernel - part of a operating system, resides in memory at all times on the computer,
- others
  - system programs - all programs associated with the operating system, but not in kernel
  - application program - all programs not associated with the operating system
  - middleware – a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics

Kernel in Memory



Operating System Is Interrupt-Driven



- Device driver (software) inside the kernel knows how to talk and manage the device
- Device driver provides uniform interface between controller and kernel
- Each device controller informs CPU that it has finished its operation by causing an interrupt

Interrupts

- An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention
  - Software: A trap or exception is a software-generated interrupt caused either by an error or a user request (system call)
  - Hardware: Device
- ISR (Interrupt Service Routines) inside the kernel determine what action should be taken for each type of interrupt
- An operating system is interrupt driven
  - the code of the kernel is in memory all the time but the code of the kernel is only executed when there is an interrupt, on demand!

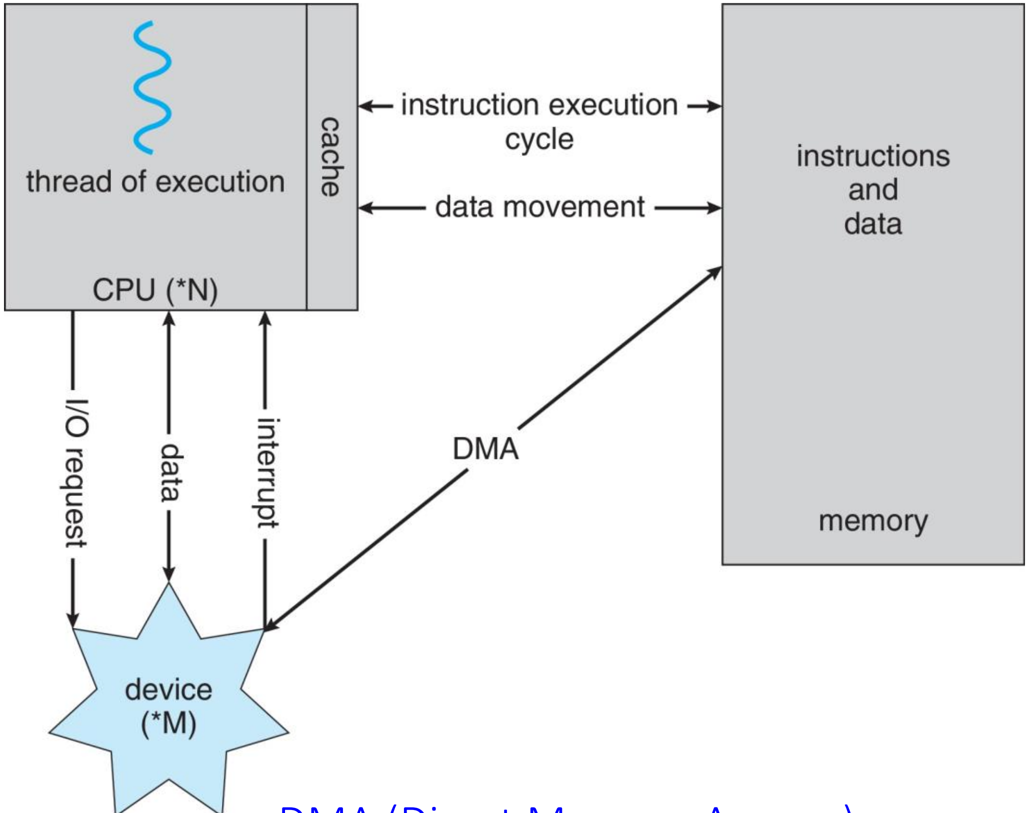
An operating system is interrupt driven

Interrupt Handling Steps

When there is an interrupt, the operating system

1. Preserves the state of the CPU by storing registers and the program counter(PC) for the software that was just interrupted (so that the same software can be restarted later).
2. Determine type of interrupt
   - check Interrupt vector to get the address of corresponding ISR for this interrupt (used on all modern computers)
3. Runs ISR in kernel to handle interrupt

How a Modern Computer Works: DMA

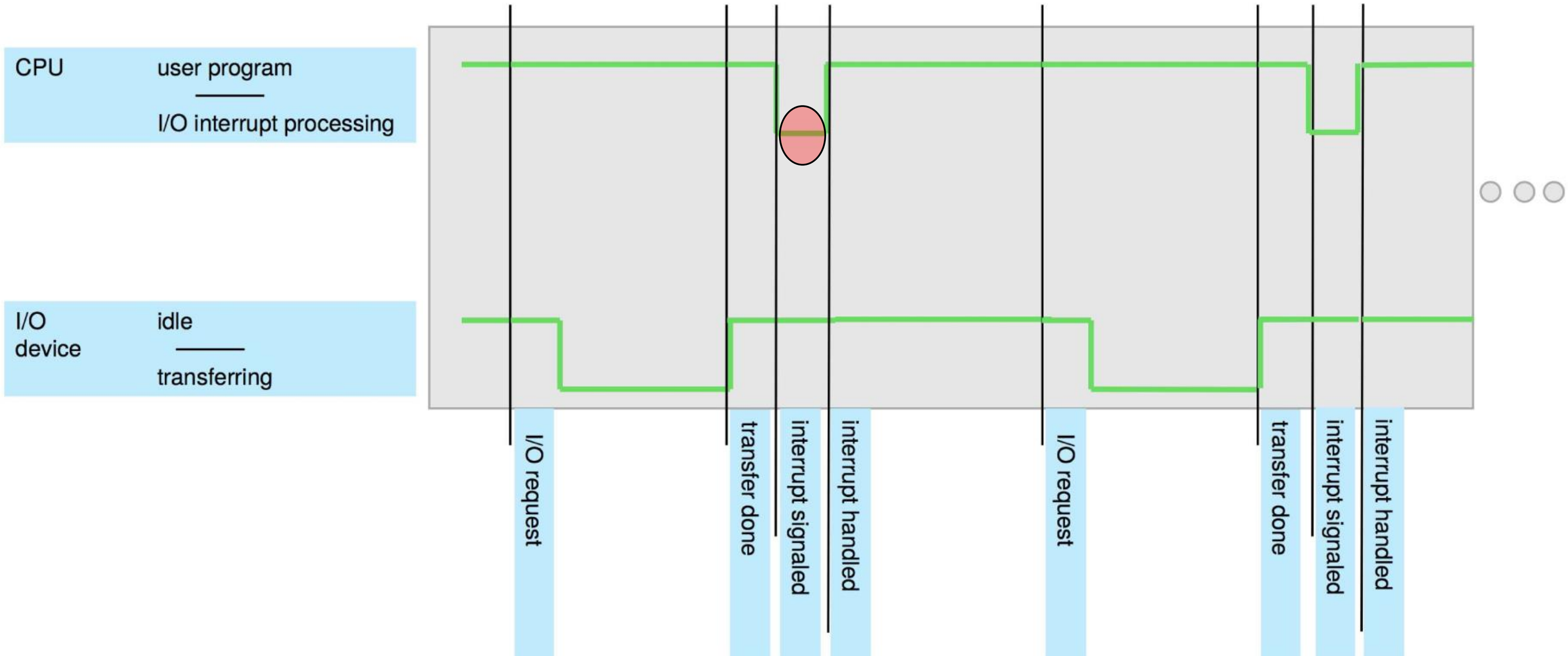


A von Neumann architecture

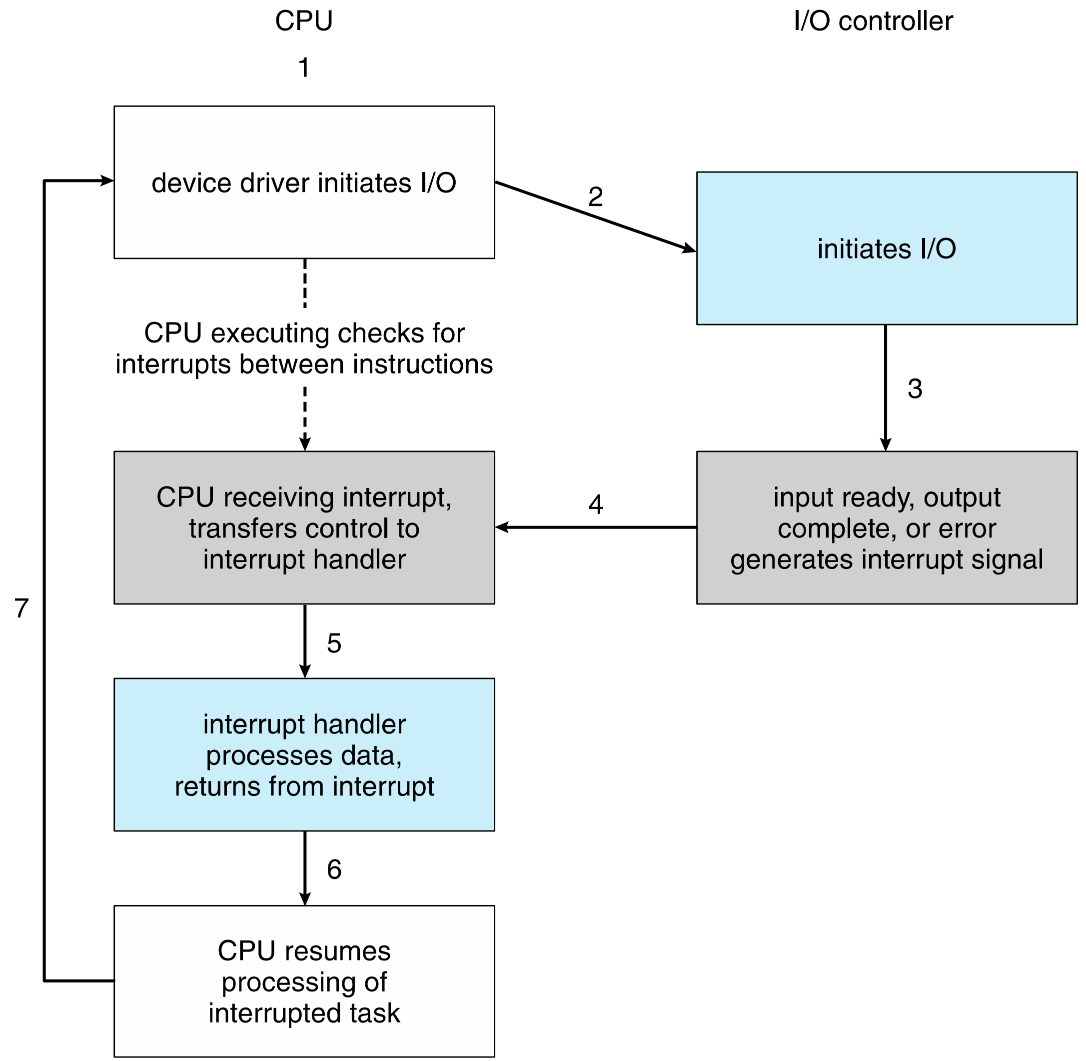
DMA (Direct Memory Access)

- Device controller transfers blocks of data from buffer directly to main memory without CPU intervention
  - In the meantime the CPU can work on something else.
- Only one interrupt is generated per block
- High transfer speed : not byte by byte, but block by block

Interrupt-Driven I/O Cycle: Timeline View



Interrupt-Driven I/O Cycle – Workflow View



Interrupt-Driven I/O Cycle: Two Methods

After I/O starts, control returns to user program only upon I/O completion

After I/O starts, control returns to user program without waiting for I/O completion

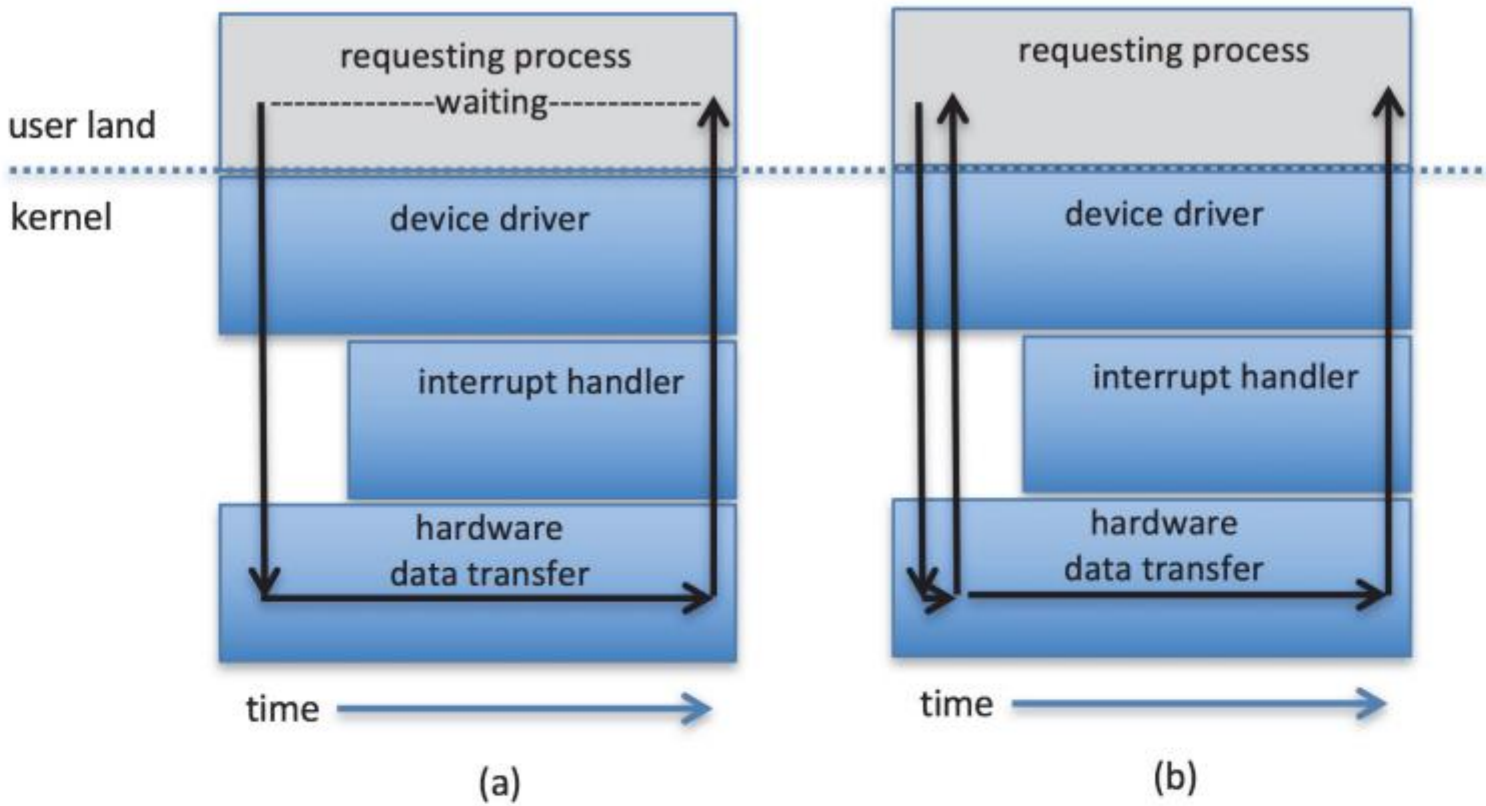


Figure 12.9 Two I/O methods: (a) synchronous and (b) asynchronous.

Operating System Activities

Operating system does a lot of management work

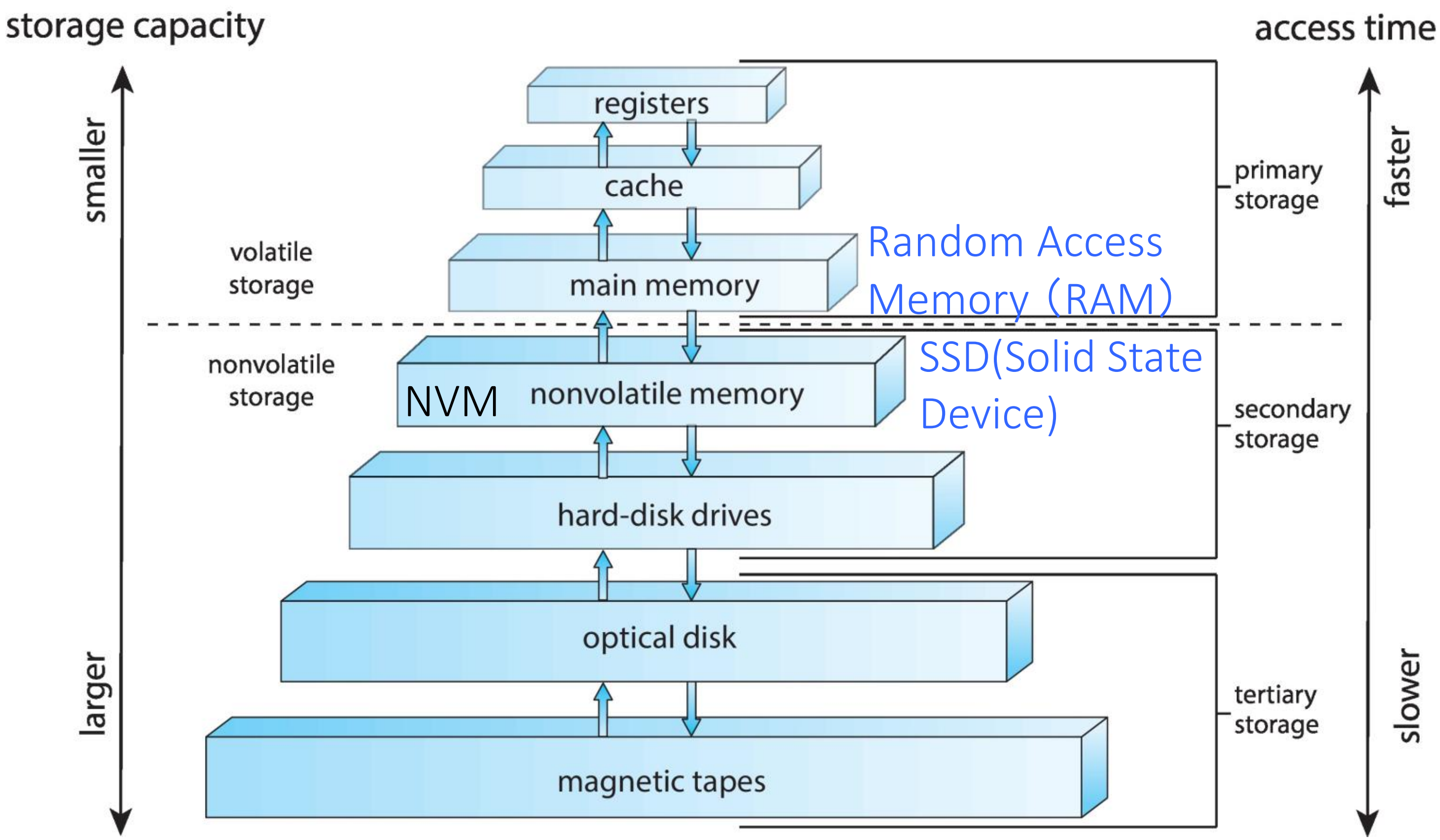
- Caching
- Process management
- Memory management
- File management
- Secondary storage management
- I/O
- Protection and security

Operating System Activities

Operating system does a lot of management work

- Caching
- Process management
- Memory management
- File management
- Secondary storage management
- I/O
- Protection and security

Storage-Device Hierarchy



Comparison:

- Speed
- Cost
- Volatility

Storage Size Units

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; and 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 Size Units（In Chinese）

中文单位	中文简称	英文单位	英文简称	进率（Byte=1）
位元	比特	bit	b	0.125
字节	字节	Byte	B	1
千字节	千字节	KiloByte	KB	2^10
兆字节	兆	MegaByte	MB	2^20
吉字节	吉	GigaByte	GB	2^30
太字节	太	TeraByte	TB	2^40
拍字节	拍	PetaByte	PB	2^50
艾字节	艾	ExaByte	EB	2^60
泽字节	泽	ZettaByte	ZB	2^70
尧字节	尧	YottaByte	YB	2^80
珀字节	珀	BrontoByte	BB	2^90

Characteristics of Various Types of Storage

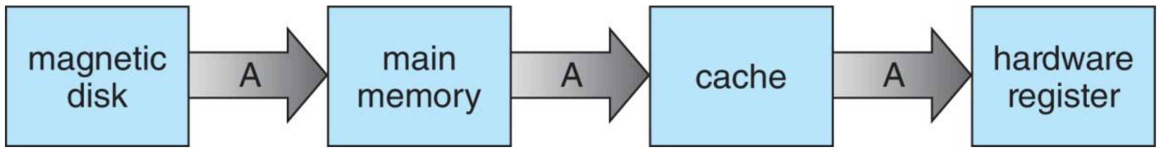
Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	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

Data moves between levels of storage hierarchy explicitly or implicitly

Caching

Caching： Copy information from slower into faster storage system

Purpose： Speed up data access



Speed： slow → fast

Size： big → small

Price： cheap → expensive

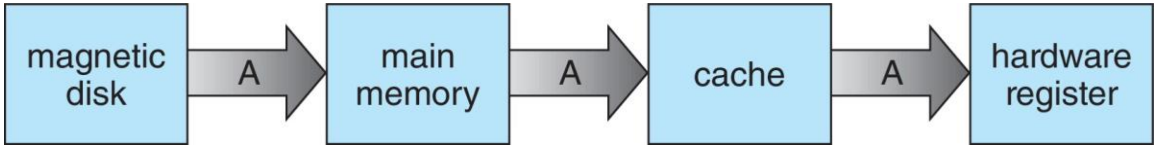
- Data access procedure： faster storage (cache) checked first
  - Data exist, information is used directly from the cache (fast)
  - Data not exist , data are copied from slower device to the faster.

Cache design consideration

size

content replacement policy

Cache Coherency



Migration of data "A" from Disk to Register

Cache coherency: The coordination of the contents of caches such that an update to a value stored in one cache is immediately reflected in all other caches that hold that value

Operating system activities

Operating system does a lot of management work

- Caching
- Process management
- Memory management
- File management
- Secondary storage management
- I/O
- Protection and security

Process

- Program vs process
  - A process is a program in execution
    - Program is a passive entity, on storage
    - Process is an active entity, in memory
- Process
  - creation, execution needs resources
    - CPU, memory, I/O, files
    - Initialization data
  - Termination requires reclaim of any reusable resources
- Process can be
  - single-threaded (线程), or
  - multi-threaded
- Each process has a program counter to
  - specify location of next instruction to execute

Multiprogramming and Multitasking

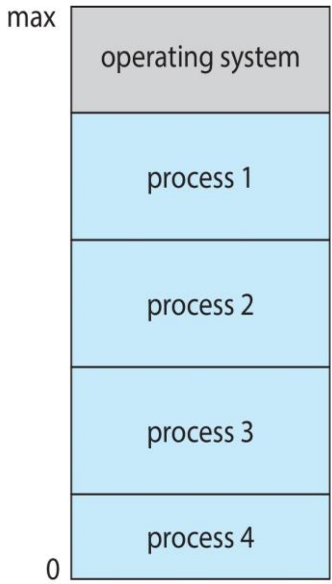
Single process cannot always keep CPU and I/O devices busy

Multiprogramming : A subset of total jobs are kept in memory at the same time

- Job scheduling: choose which jobs to load into memory.
- When job has to wait (for I/O for example), OS switches to another job

Multitasking: The CPU switches jobs so frequently to increase interactions with users

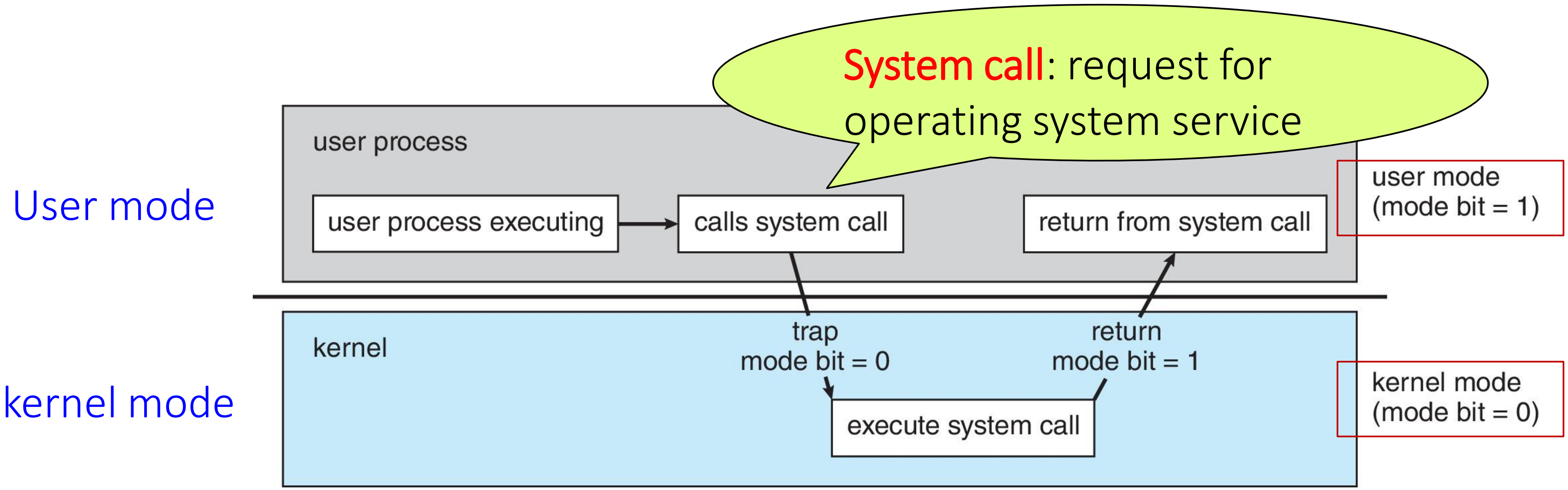
- CPU scheduling: choose which job will run next If several jobs are ready to run at the same time



Job scheduling CPU scheduling

Secondary storage ----------> memory -----------> CPU

Process Execution: Dual-mode



Dual-mode operation allows OS to protect itself and other system components

Process Management Activities

Process management activities include:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Process synchronization
- Process communication
- Deadlock handling

Operating system activities

Operating system does a lot of management work

- Caching
- Process management
- Memory management
- File management
- Secondary storage management
- I/O
- Protection and security

Memory Management

To execute a program,

all (or part) of the instructions of the program must be in memory

All (or part) of the data needed by the program must be in memory

Memory management determines what is in memory and when

Memory management activities include

1. Keeping track of which parts of memory are currently being used and by whom
2. Deciding which processes and data (or part of them) to move into and out of memory
3. Allocating and de-allocating memory space as needed

Operating system activities

Operating system does a lot of management work

- Caching
- Process management
- Memory management
- File management
- Secondary storage management
- I/O
- Protection and security

File System Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Files are usually organized into directories
  - Access control determines who can access what
  - File system activities include
    1. Creating and deleting files and directories
    2. Mapping files onto secondary storage

Secondary Storage Management

- Disks usually are used to store data that do 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
- Secondary storage management activities
  1. Mounting and unmounting
  2. Free-space management
  3. Storage allocation
  4. Disk scheduling
  5. Partitioning
  6. Protection

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including 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)
  - General device-driver interface
  - Drivers for specific hardware devices

Spooling: Simultaneous Peripheral Operations Online

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
- 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
  - Group identifier (group ID) allows set of users to be defined

Can A Compiled Program Be Portable

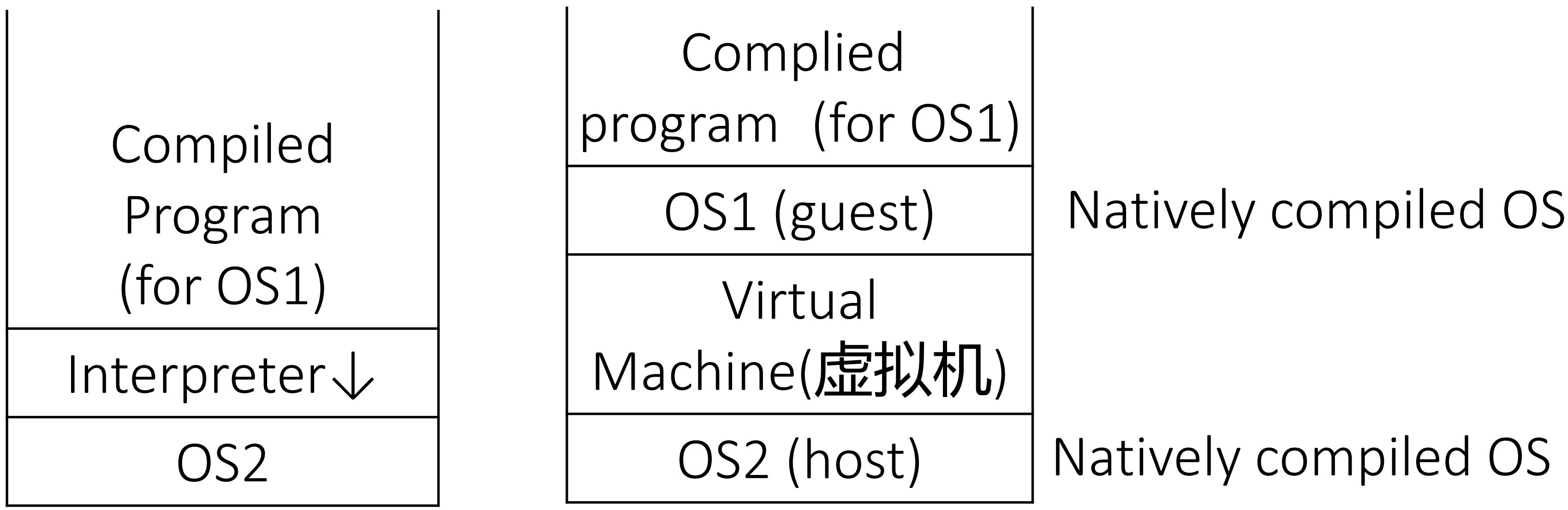
Compiled Program → OS1 (for the CPU in that computer)

Compiled Program → ? OS2 (for another CPU in another computer)

Solutions?

Virtualization

Solutions: two methods.



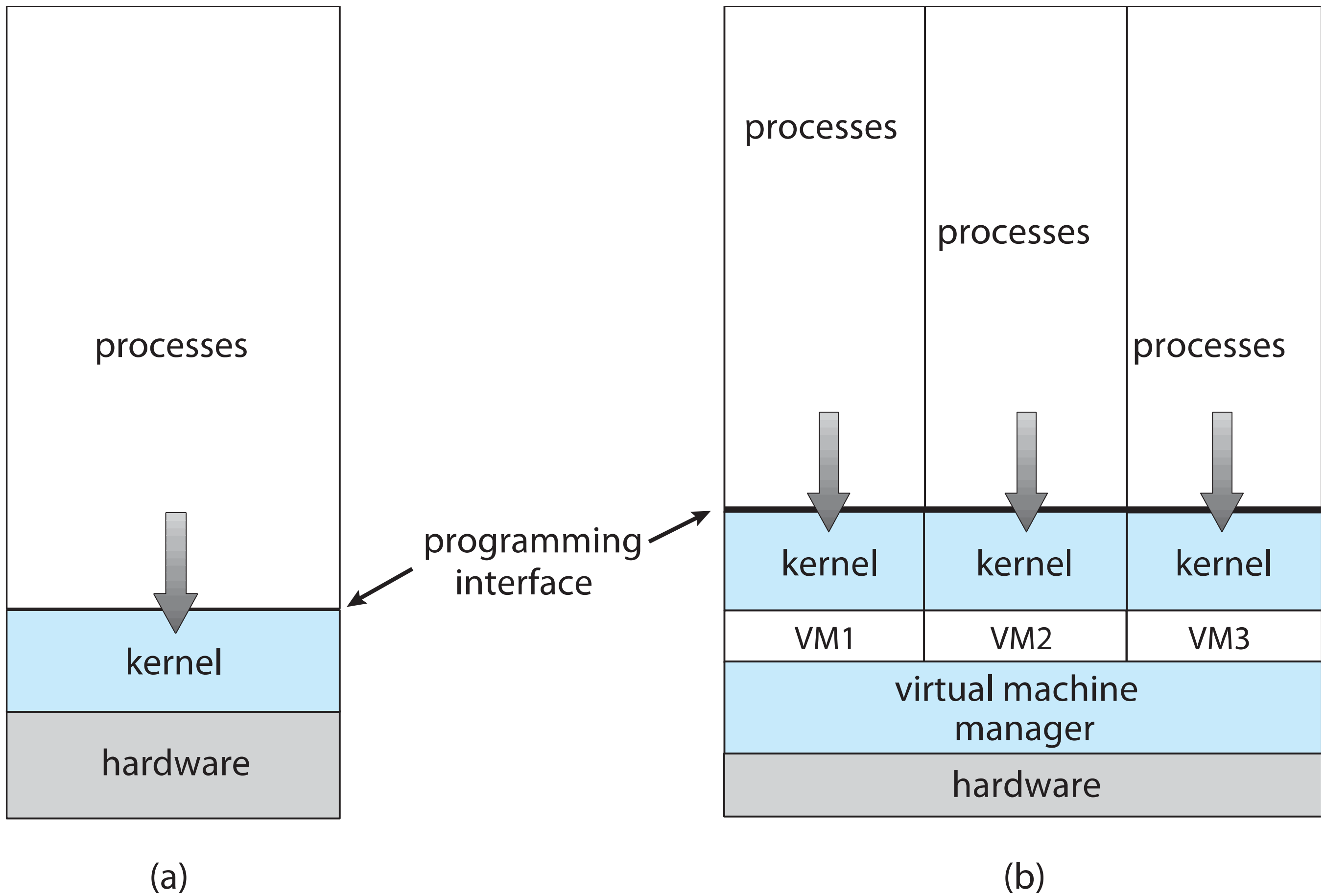
Emulation (Old method)

Virtualization

Virtual Machine: VirtualBox, VMware

Virtualization is a technology that allows us to abstract the hardware of a single computer into several different execution environments, thereby creating the illusion that each separate environment is running on its own private computer.

Virtualization



VMMs (Virtual Machine Managers)

- Run natively, no longer run on host operating systems but rather *are* the host operating systems, providing services and resource management to virtual machine processes.
- Examples: VMware ESX and Citrix XenServer

Virtualization Examples

Use cases

- Apple laptop running Mac OS X host, Windows as a guest
- Developing apps for multiple OSes without having multiple systems
- Quality assurance testing applications without having multiple systems
- Executing and managing compute environments within data centers

After class:

Virtualization vs. Dual OS ???

Free and Open-Source Operating Systems

Practical differences between free software (social movement) and open source (development technology)

- all existing released free software source code would qualify as open source.
- Nearly all open source software is free software, but there are exceptions.
- More differences see web https://www.geeksforgeeks.org/difference-between-free-software-and-open-source-software/

Free and Open-Source Operating Systems

Free or open source operating systems are made available in source-code format rather than just binary closed-source and proprietary

Examples

- GNU/Linux and BSD UNIX (including core of Mac OS X), and many more
- VM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)

The Study of Operating Systems

There has never been a more interesting time to study operating systems, and it has never been easier. The open-source movement has overtaken operating systems, causing many of them to be made available in both source and binary (executable) format. The list of operating systems available in both formats includes Linux, BUSD UNIX, Solaris, and part of macOS.
The availability of source code allows us to study operating systems from the inside out. Questions that we could once answer only by looking at documentation or the behavior of an operating system we can now answer by examining the code itself.

Operating systems that are no longer commercially viable have been open-sourced as well, enabling us to study how systems operated in a time of fewer CPU, memory, and storage resources.
An extensive but incomplete list of open-source operating-system projects is available from https://curlie.org/Computers/Software/Operating_Systems/Open_Source/

In addition, the rise of virtualization as a mainstream (and frequently free) computer function makes it possible to run many operating systems on top of one core system. For example, VMware (http://www.vmware.com) provides a free "player" for Windows on which hundreds of free "virtual appliances" can run. Virtualbox (http://www.virtualbox.com) provides a free, open-source virtual machine manager on many operating systems. Using such tools, students can try out hundreds of operating systems without dedicated hardware.

The advent of open-source operating systems has also made it easier to make the move from student to operating-system developer. With some knowledge, some effort, and an Internet connection, a student can even create a new operating-system distribution. Just a few years ago, it was difficult or impossible to get access to source code. Now, such access is limited only by how much interest, time, and disk space a student has.

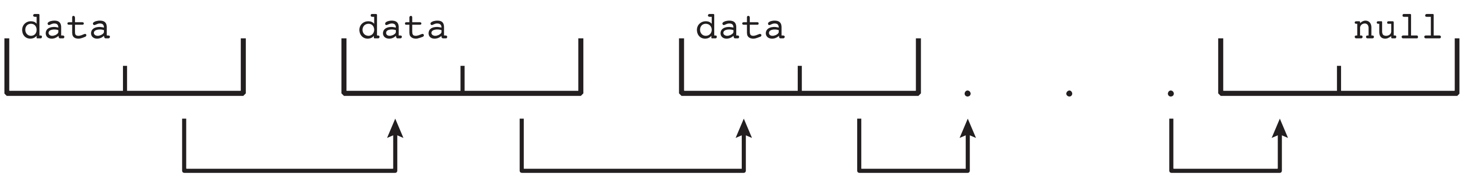
Free and Open-Source Operating Systems

- Operating system used in this semester
  - Ubuntu
- Three ways to access Ubuntu in this semester
  - Install it in your own computer following guidelines in iSpace (Virtual machine: Virtualbox)
    - Virtualbox download: https://www.virtualbox.org/wiki/Downloads
  - Access computers in CST lab rooms (Virtual machine: VMware)
  - Access through the network (ssh to server with user account and password).

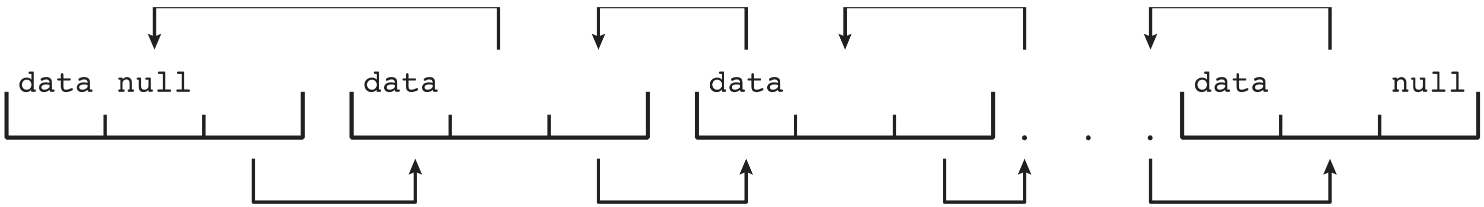
Kernel Data Structures

Many similar to standard programming data structures

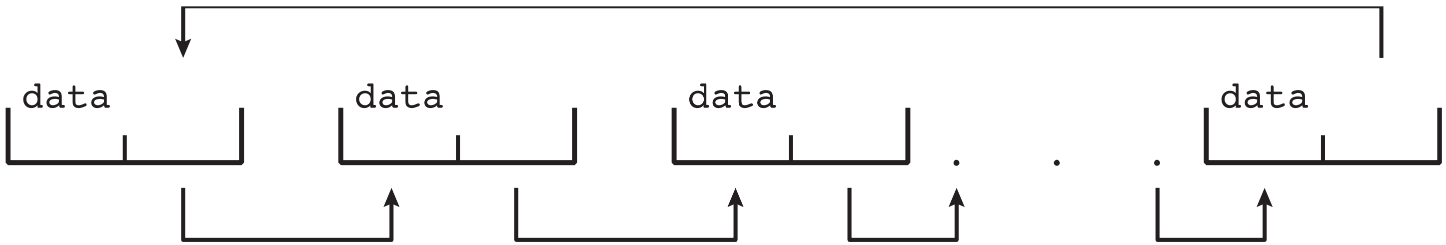
Singly linked list



Doubly linked list



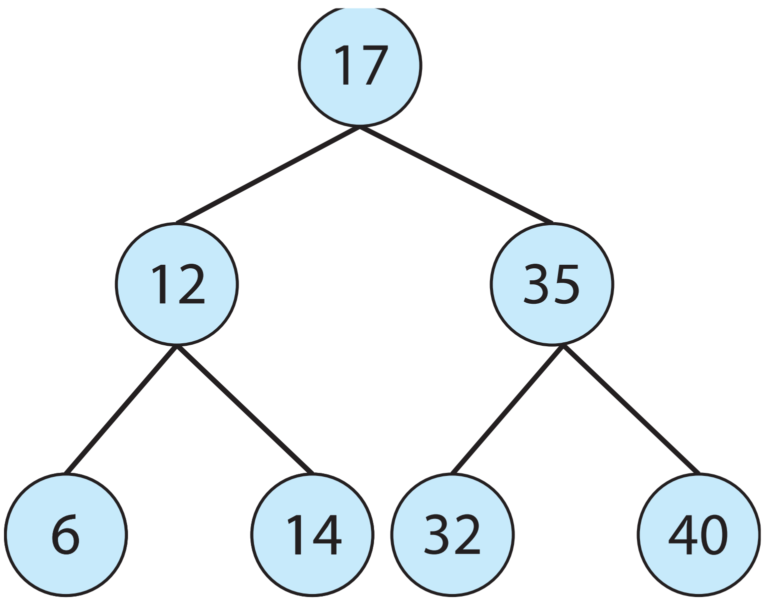
Circular linked list



Kernel Data Structures

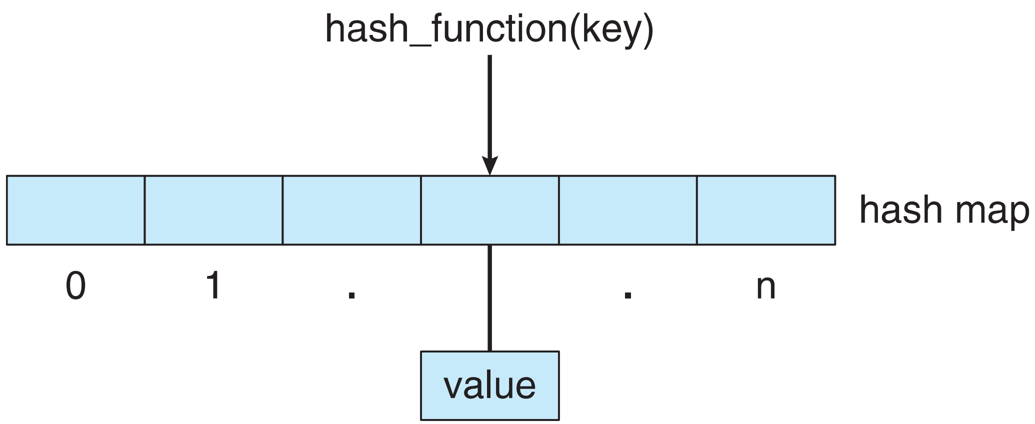
Binary search tree

left <= right



Kernel Data Structures

Hash function can create a hash map



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>

End of Chapter 1