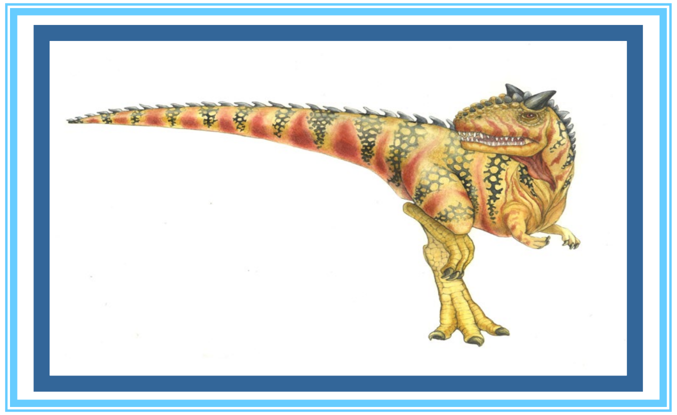
Chapter 1: Introduction



Objectives

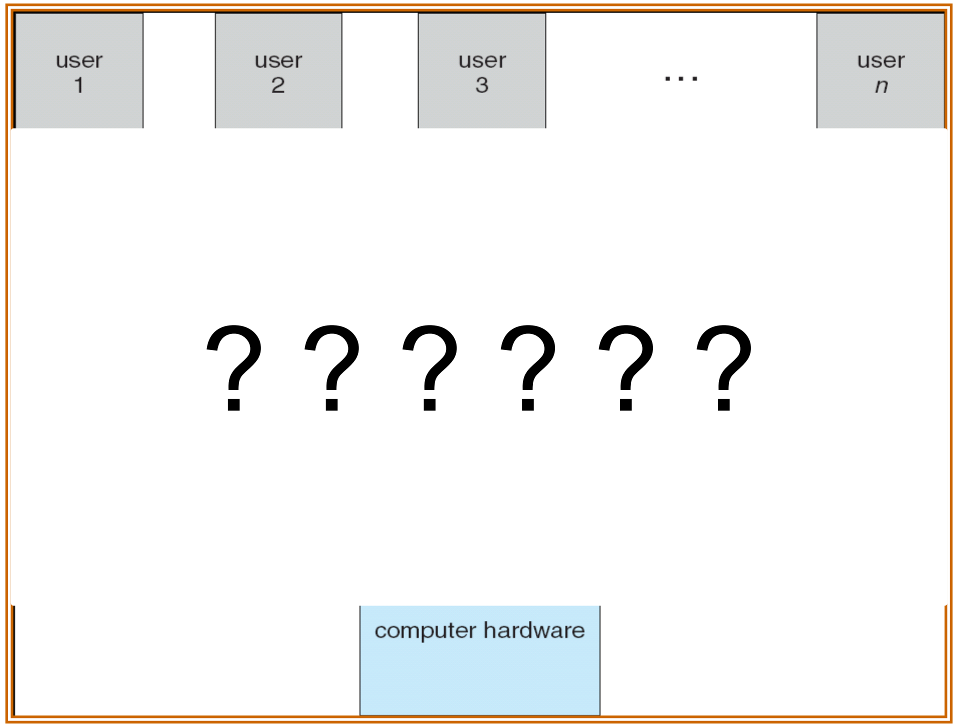
- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

Chapter 1: Introduction

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

What is an Operating System?

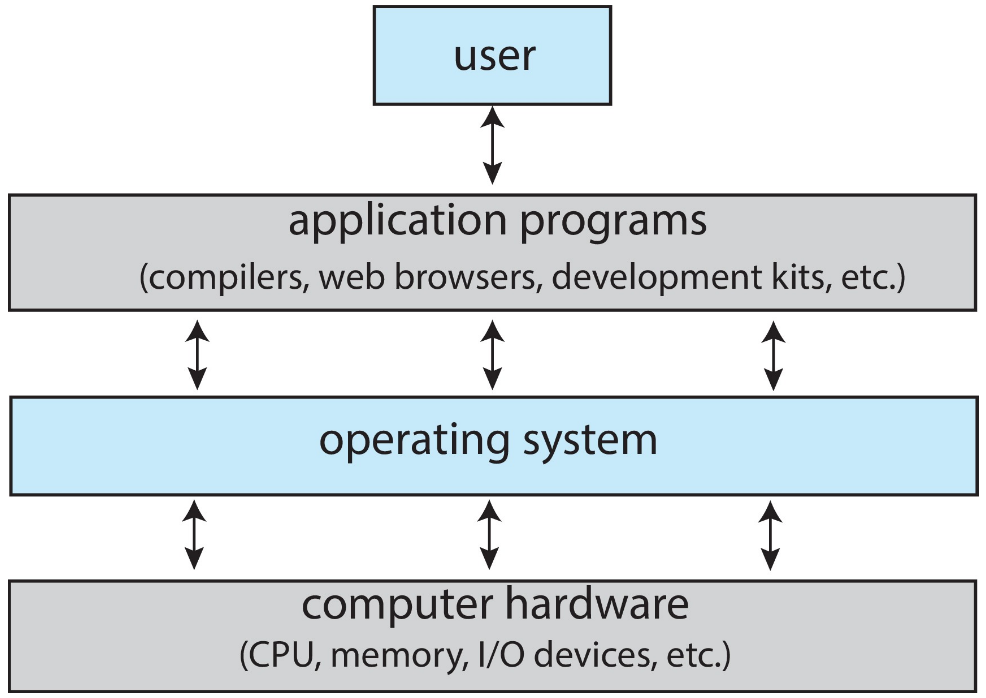
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner



Computer System Structure

- 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

Four Components of a Computer System


user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

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**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont.)

Poll 1

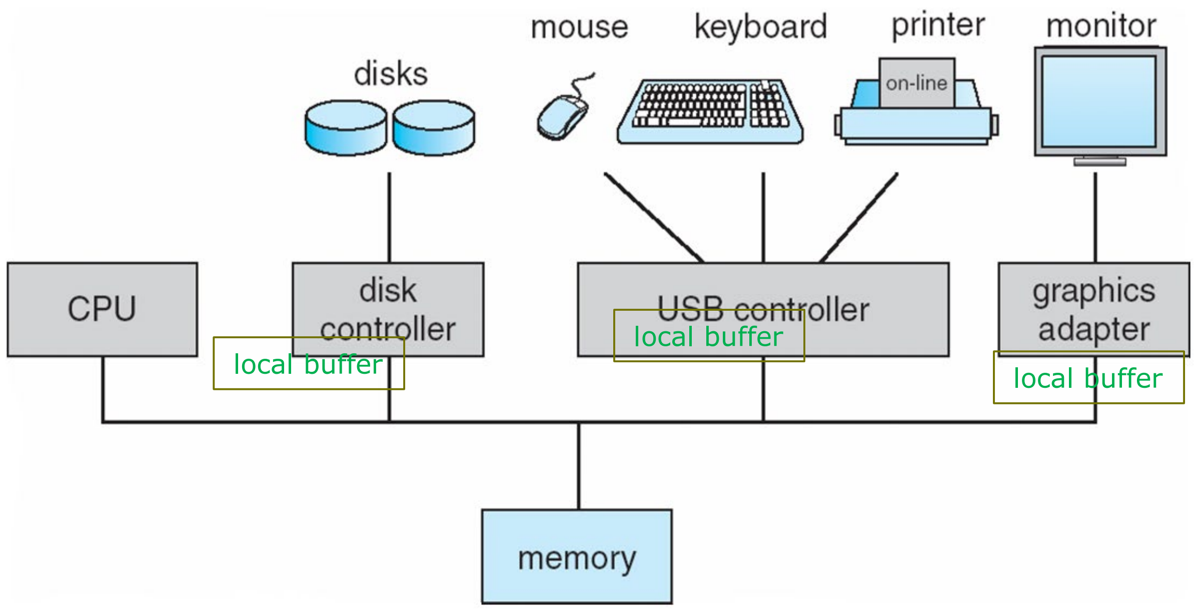
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly

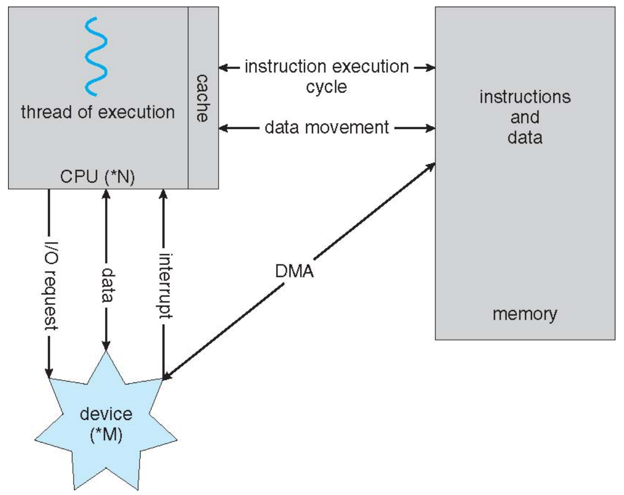
As an CS300 student, an answer must be different and better …

- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.
- Today's OSes for general purpose and mobile computing also include ***middleware*** – a set of software frameworks that provide addition services to application developers such as databases, multimedia, graphics

Computer System Organization

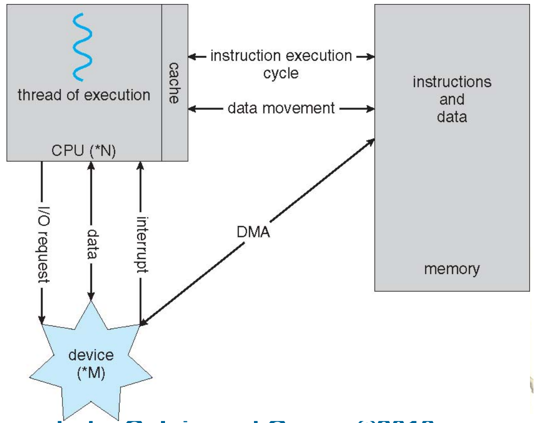
- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
    - Direct Memory Access
  - Concurrent execution of CPUs and devices competing for memory cycles





Computer-System Operation

- 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
- 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
- For DMA
  - 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
  - Only one interrupt is generated per block, rather than the one interrupt per byte



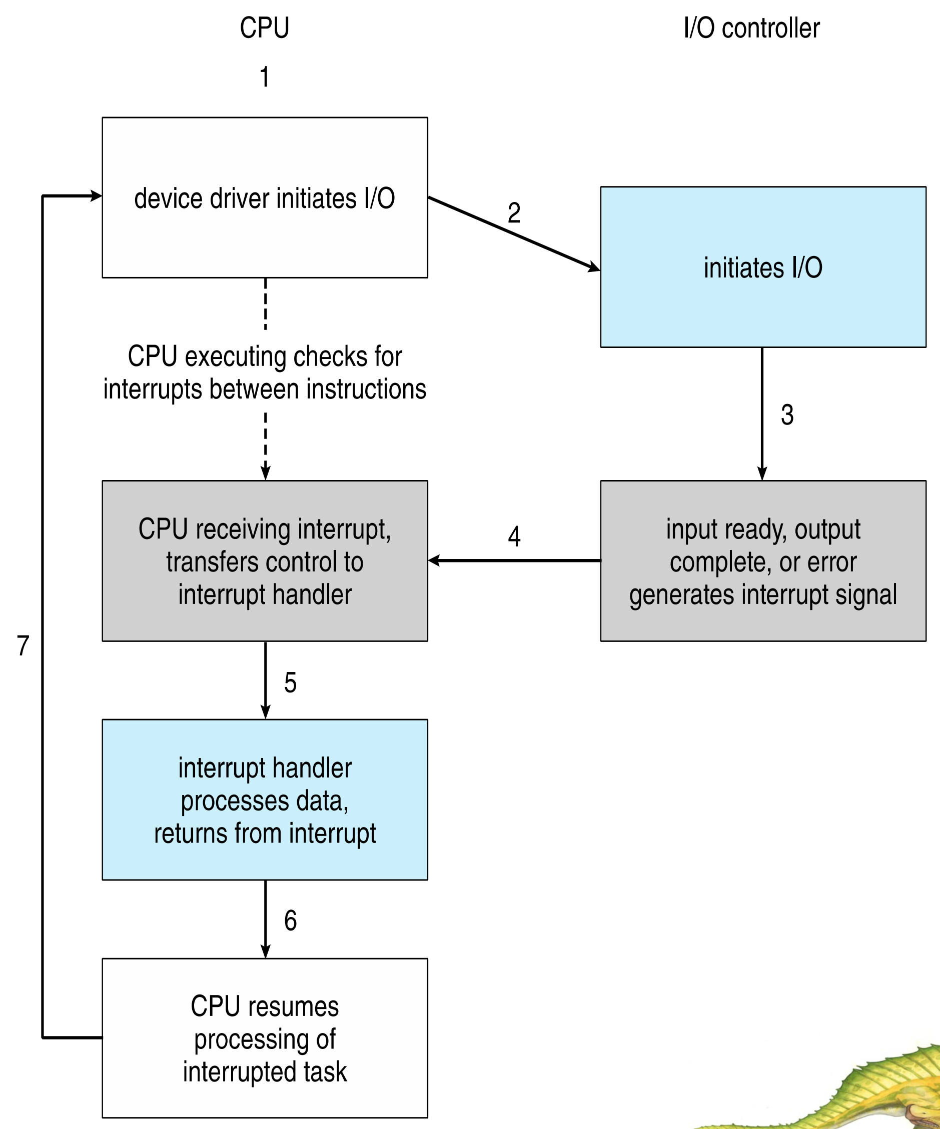
Common Functions of Interrupts

- 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
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Causes changes of the program execution

Interrupt Handling

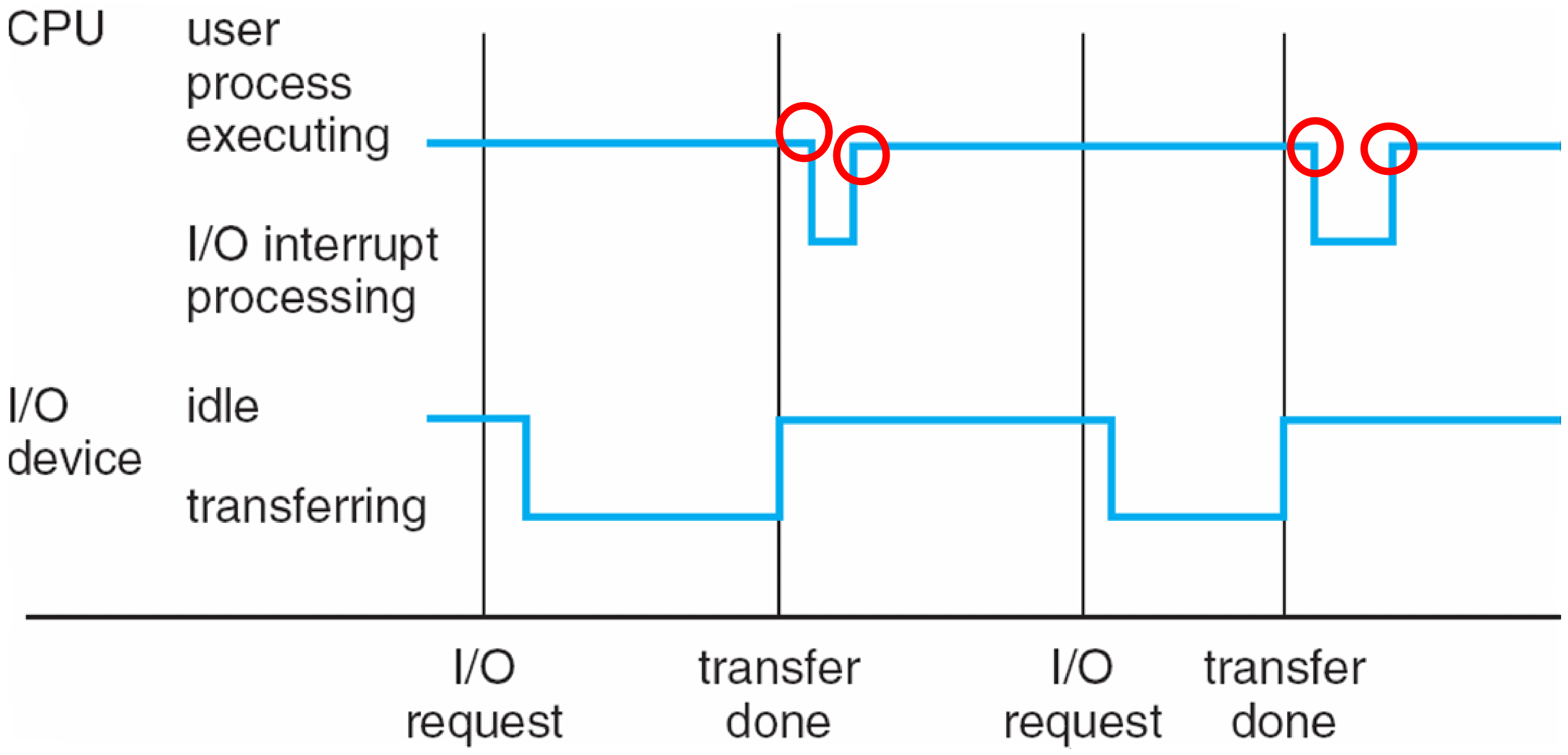
- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred
- Separate segments of code determine what action should be taken for each type of interrupt



CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing 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

Interrupt Timeline

What happens at these locations?



I/O handling

Poll 2

- Method1: 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

Blocked behavior

- Method 2: After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS to allow user to wake up for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Non-Blocked behavior

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).

Remember this!
K = 2^10;
M = K*K = 2^20;
G = 2^30;
…

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access,** Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity. Some are:

- 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

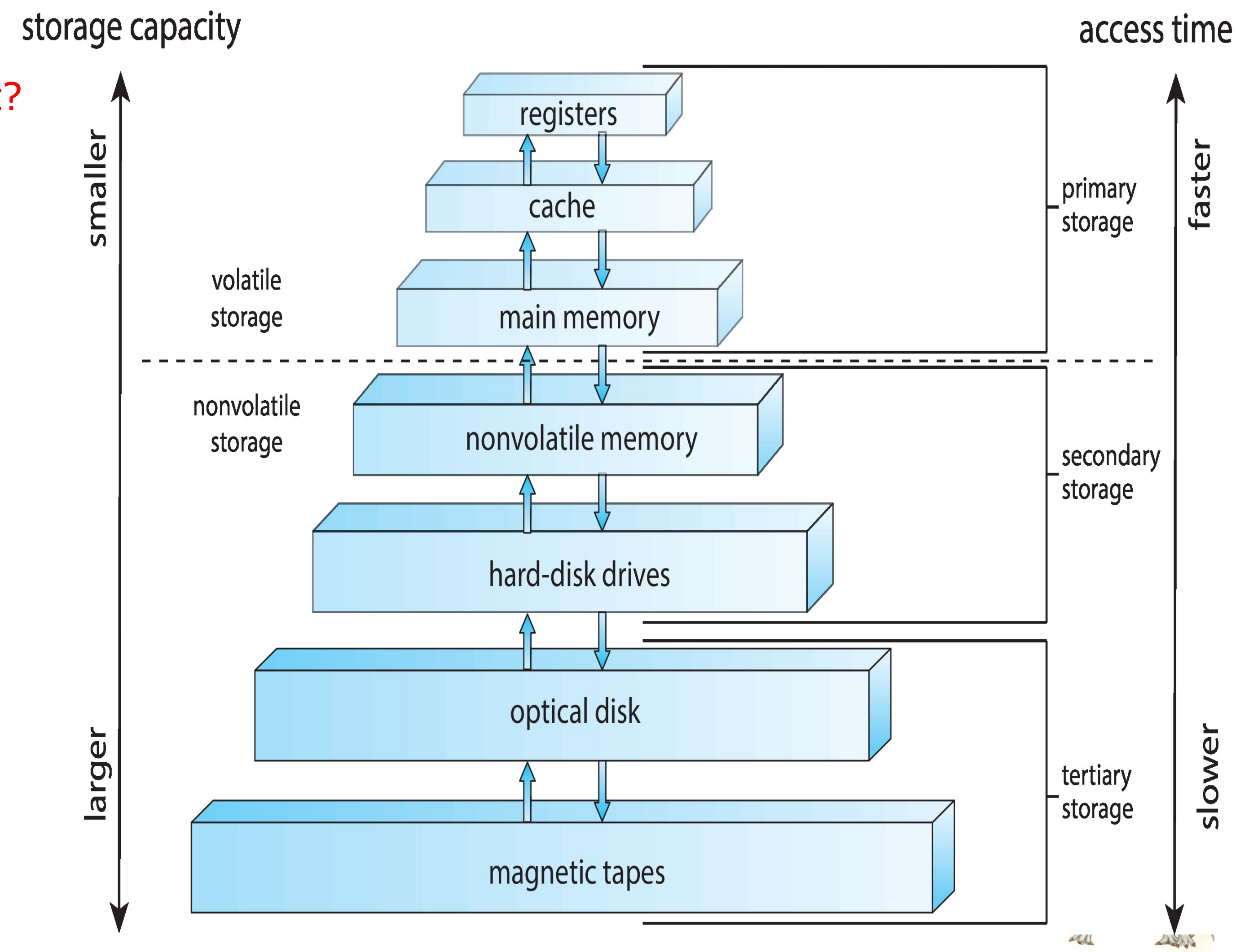
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

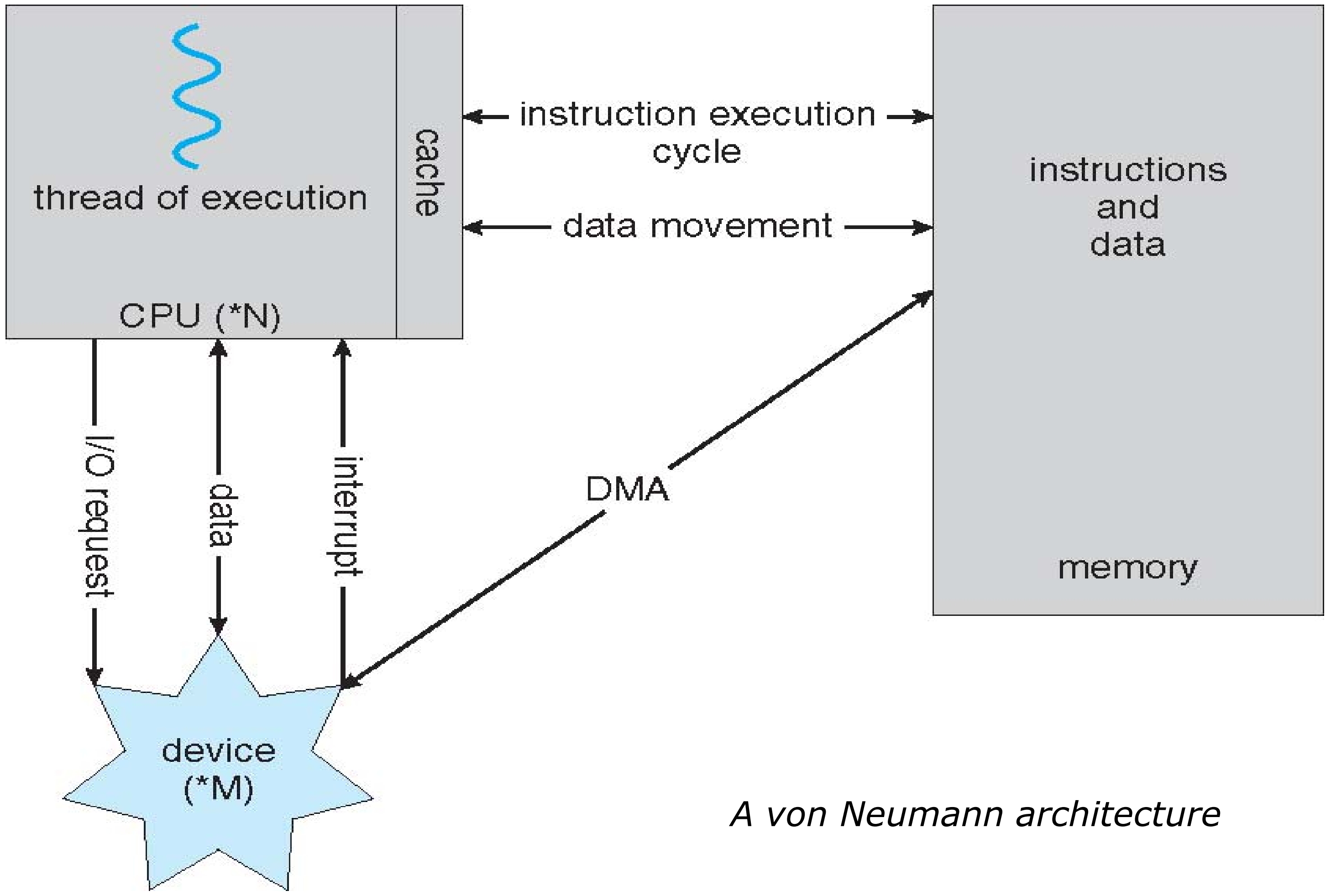
Which matters the most?

- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-Device Hierarchy



How a Modern Computer Works



A von Neumann architecture

Chapter 1: Introduction

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

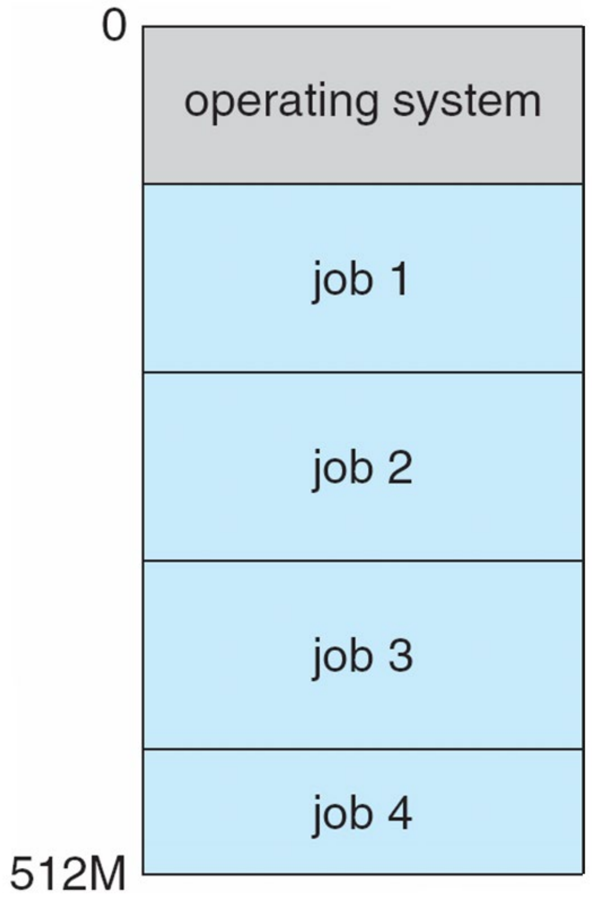
Operating-System Operations

- Bootstrap program – simple code to initialize the system, load the kernel
- Kernel loads
- Starts **system daemons** (services provided outside of the kernel)
- Kernel **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service – **system call**
    - Other process problems include infinite loop, processes modifying each other or the operating system

Multiprogramming (Batch system)

- **Multiprogramming** (**Batch system**) needed for efficiency
  - 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
  - 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

Memory Layout for Multiprogrammed System

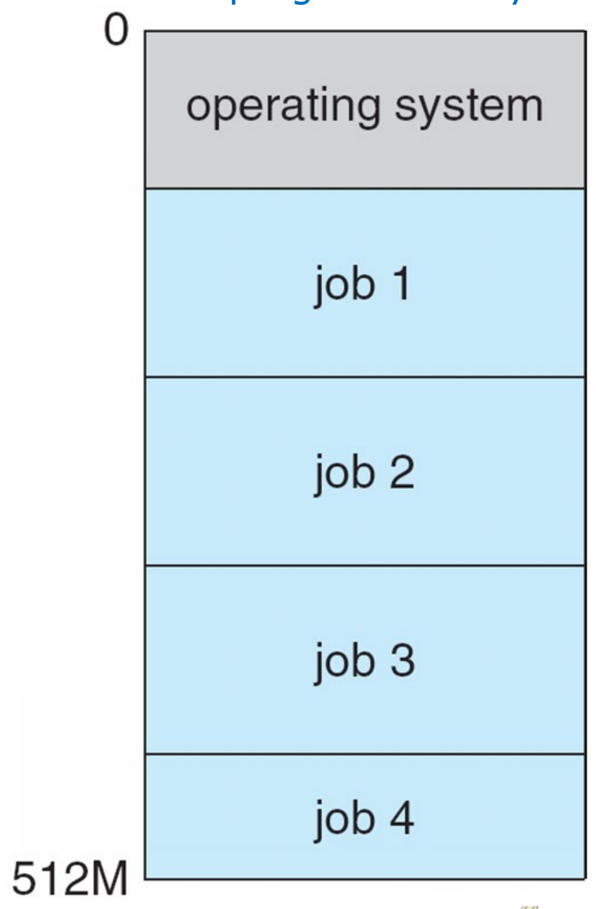


Multitasking (Timesharing)

Poll 3

- **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

Memory Layout for Multiprogrammed System



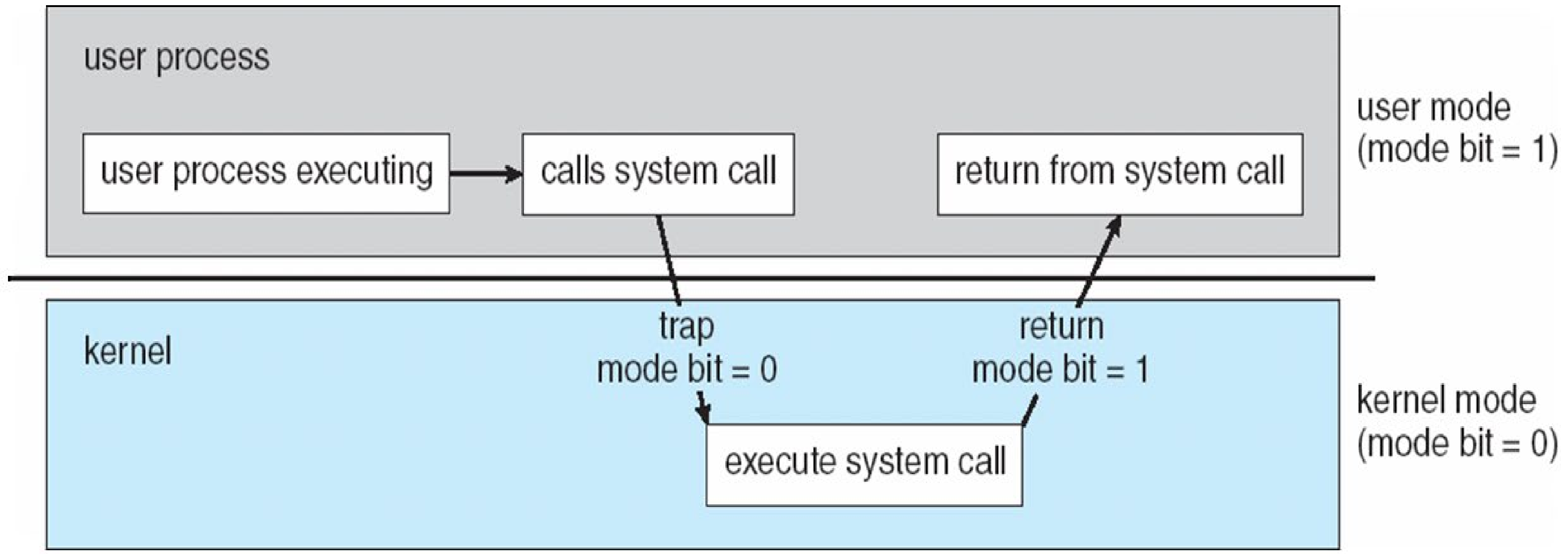
Dual-mode Operation

Example of OS, a software, being supported by certain hardware. Keep an eye on other examples…

Hardware Protection

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
      - When a user is running ⇨ mode bit is "user"
      - When kernel code is executing ⇨ mode bit is "kernel"
- How do we guarantee that user does not explicitly set the mode bit to "kernel"?
    - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as **privileged**, only executable in kernel mode

Transition from User to Kernel Mode



Which mode is in at the boot time?

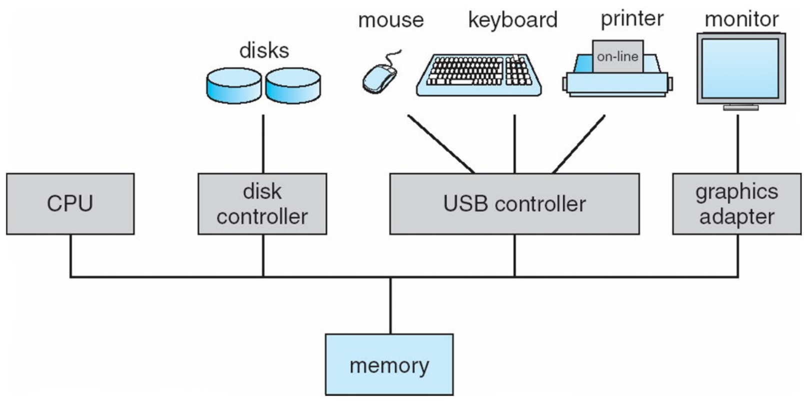
Timer

Another piece of hardware

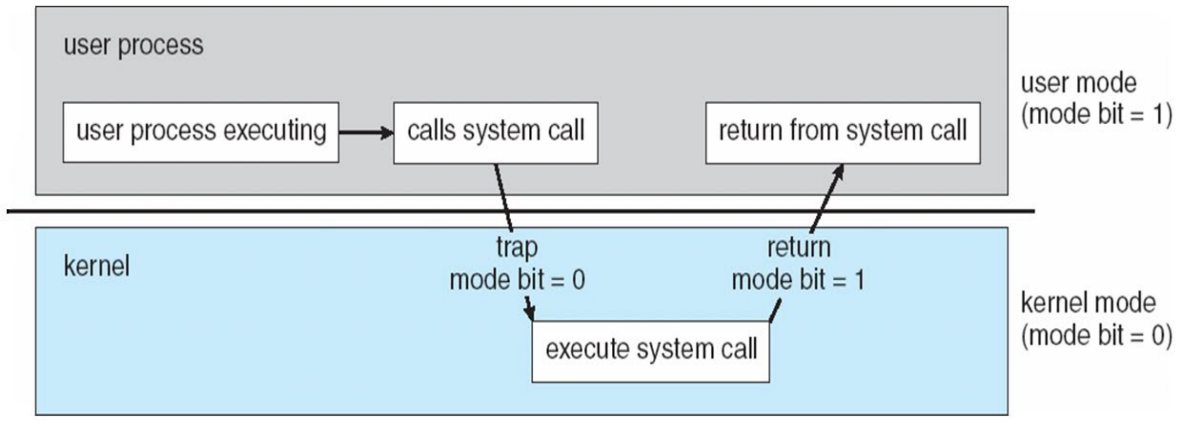
- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up timer before scheduling process to regain control or terminate program that exceeds allotted time

Recap

- Computer system



- OS is a **resource allocator,** a **control program**
- **Multiprogramming and Timesharing** (**multitasking**)
- **Interrupt driven** (hardware and software)
- **Dual-mode** operation allows OS to protect itself and other system components
- **Timer**



Chapter 1: Introduction

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

Process Management

- 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 users, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

OS as a resource allocator

OS as a control program

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

OS as a control program

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
  - 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

Storage Management

- 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

- 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

- 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
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

Caching, Why and How

- 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 is smaller than the storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

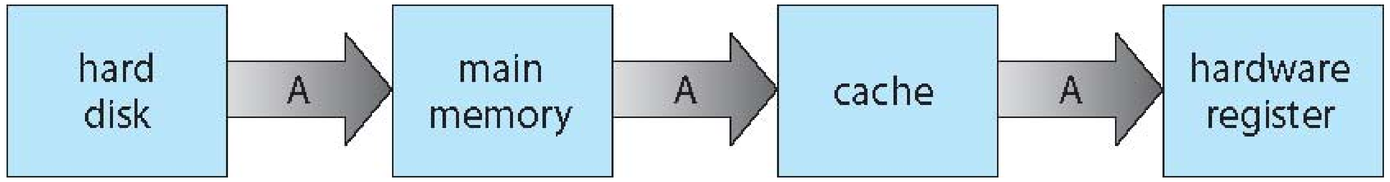
Characteristics of Various Types of Storage

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

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data “A” from Disk to Register

- Movement between levels of storage hierarchy can be explicit or implicit
- Data can exist at multiple places
- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



Migration of data “A” from Disk to Register

- 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 solutions covered in Chapter 17

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

Chapter 1: Introduction

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

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

Protection and Security

- 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 (**group 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
- Protection examples:
  - memory-addressing hardware ensures that a process can execute only within its own address space
  - timer ensures that no process can gain control of the CPU without eventually relinquishing control

Protection and Security

- Security:
  - Prevention of some of these attacks is considered an operating-system function
  - And use policy or additional software.

Side note: What functionality goes into the OS?

- as much as necessary, as little as possible
  - OS code is very expensive to develop and maintain

- functionality must be in the OS if it:
  - requires the use of privileged instructions
  - requires the manipulation of OS data structures
  - must maintain security, trust, or resource integrity

- functions should be implemented in libraries if it:
  - is a service commonly needed by applications
  - does not actually **have to** be implemented inside OS

Summary

- The basic organization of computer systems
- An overview of the major components of operating systems

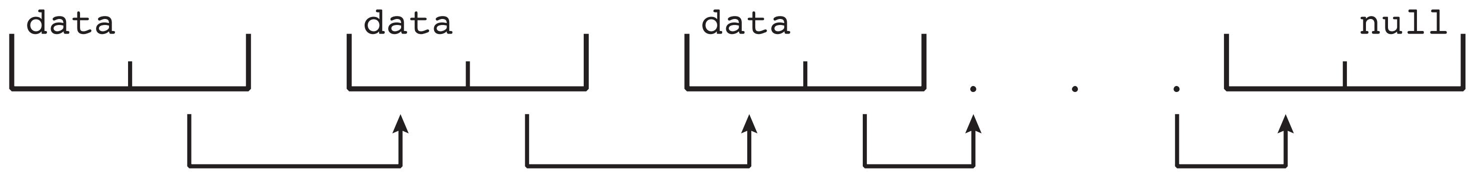
  - Familiar with C
  - Read chapters 1 and 2.

Chapter 1: Introduction

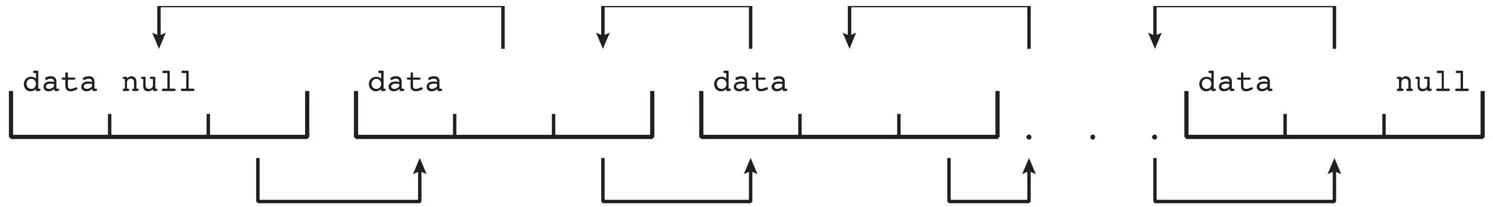
- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
  - Process Management
  - Memory Management
  - Storage Management
- Protection and Security
- Virtualization
- Distributed Systems
- Kernel Data Structures
-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

Kernel Data Structures

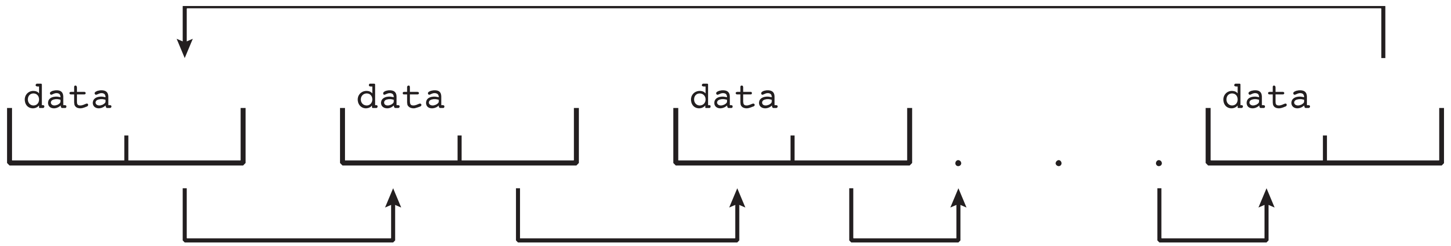
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

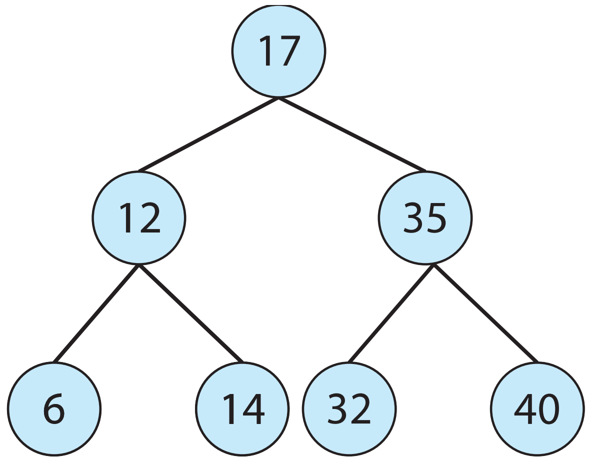


- ***Circular linked list***



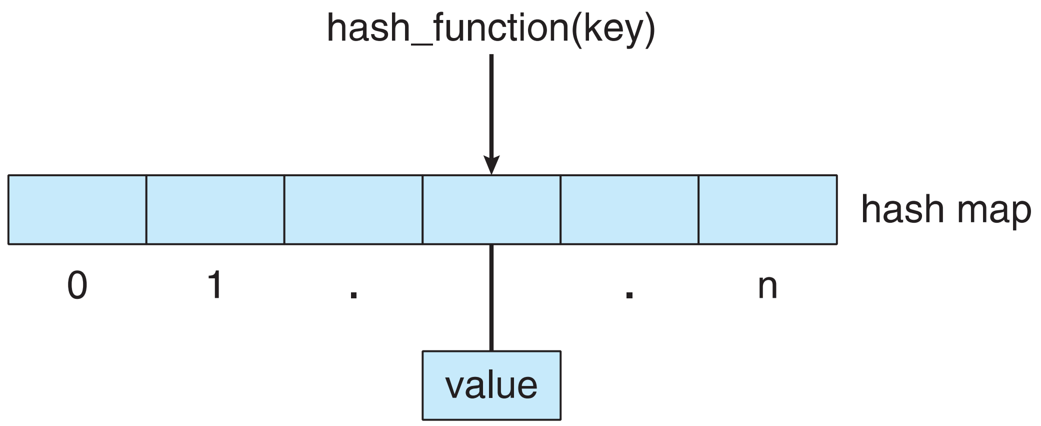
Kernel Data Structures

- **More powerful data structures**
  - **Stack, queue, trees**
- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*



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