Operating Systems

Chapter 1: Introduction

Chapter 1: Introduction

Chapter Objectives

Operating systems are everywhere, from cars and home appliances that include "Internet of Things" devices, to smart phones, personal computers, enterprise computers, supercomputers, and cloud computing environments.

In this chapter, we look into a general overview of the major components of a contemporary computer system as well as the functions provided by the operating system.

Dominant Operating Systems of today

- The dominant desktop operating system is Microsoft Windows
- macOS by Apple Inc. is in second place
- Varieties of Linux are collectively in third place
- In the mobile (smartphone and tablet combined) sector, Google's Android on smartphones is dominant followed by Apple's iOS
- Linux distributions are dominant in the server and supercomputing sectors. Since November 2017, all of the world's fastest 500 supercomputers run Linux-based operating systems
- Other specialized classes of operating systems, such as embedded and real-time systems, exist for many applications.

What is an Operating System?

- An **operating system** (**OS**) is system software that manages computer’s hardware.
- OS provides a basis for application programs and acts as an intermediary between the user of a computer and the computer hardware.
- A fundamental responsibility of an operating system is to allocate the resources such as the CPU, memory, I/O devices, as well as storage. to programs.
- The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner

Role of OS in the overall Computer System

A computer system can be divided roughly into four components:

1. **User**

   People, machines, other computers

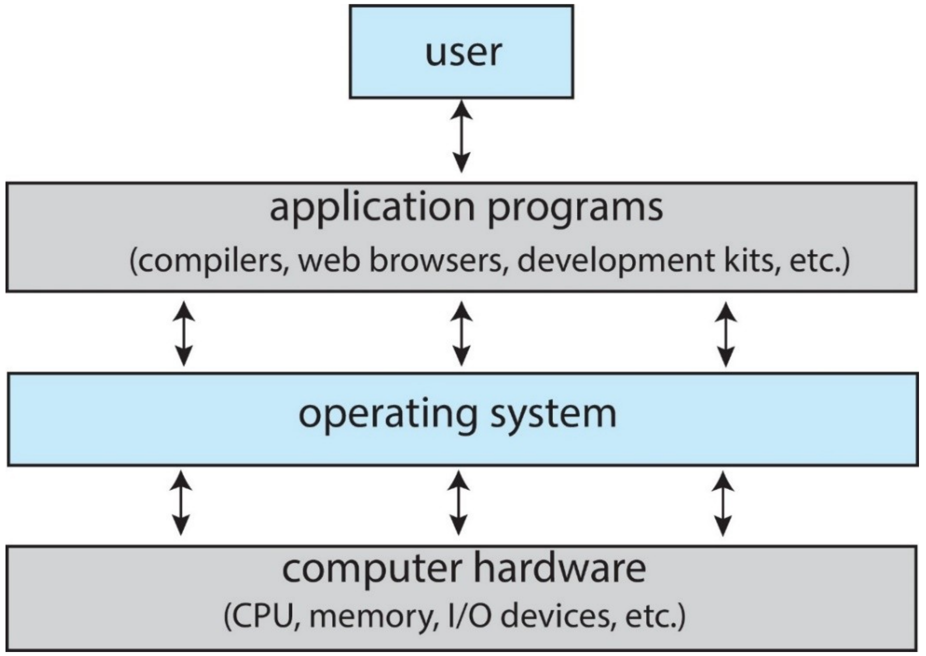
2. **Application programs** – define the ways in which the system resources are used to solve the user's computing problems

   Such as Word Processors, Compilers, Web browsers, Database systems, Video games

3. **Operating system -** controls the hardware, and coordinates its use for various application programs for the various users

4. **Hardware** – provides basic computing resources for the system

   CPU, memory, I/O devices


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

Abstract View of Components of Computer

OS design goals

Some operating systems are designed to be convenient, others to be efficient, and others to be some combination of the two:

OS from the User's Viewpoint

The user's view of the computer varies according to the interface being used:

- **Personal Computers** - single-user, no resource sharing
  - OS is designed mostly for **ease of use, security,** and **good performance**
- **Mainframe** multi-user, resource sharing
  - Operating system is a **resource allocator** and **control program,** making efficient use of hardware and managing execution of user programs
- **Mobile devices** like smartphones and tables are resource poor, optimized for usability and battery life
  - User interfaces generally feature touch screens, voice recognition
- **Embedded computers** Some computers have little or no user interface, such as embedded systems in home devices in devices and automobiles
  - operating systems and applications are designed to run primarily without user intervention

OS from the System’s viewpoint

From the computer’s point of view, the operating system is the program most intimately involved with the hardware.

1. In this context, we can view an OS as a **resource allocator**

   A computer system has many **resources** that may be required to solve a problem: **CPU time, memory space, file-storage space, I/O devices,** and so on

2. OS **controls** various **I/O devices** and user programs

3. OS **manages** the **execution** of the user programs to **prevent errors** and improper use of the computer

Defining Operating Systems

No universally accepted definition

- Some systems take up less than a megabyte of space and lack even a full-screen editor, whereas others require gigabytes of space and are based entirely on graphical windowing systems

- Mobile operating systems often include not only a core kernel but also **middleware**—a set of software frameworks that provide additional services to application developers.
- Apple’s iOS and Google’s Android—features a core kernel along with middleware that supports databases, multimedia, and graphics (to name only a few).

Defining Operating Systems (cont.)

The **one program running at all times on the computer**—usually called the **kernel**.

Along with the kernel, there are two other types of programs:

1. **system programs**, which are associated with the operating system but are **not necessarily part of the kernel,** and

2. **application programs**, which include all programs not associated with the operation of the system.

The operating system includes:

In summary, for our purposes, the operating system includes:

1. The always running **kernel**,

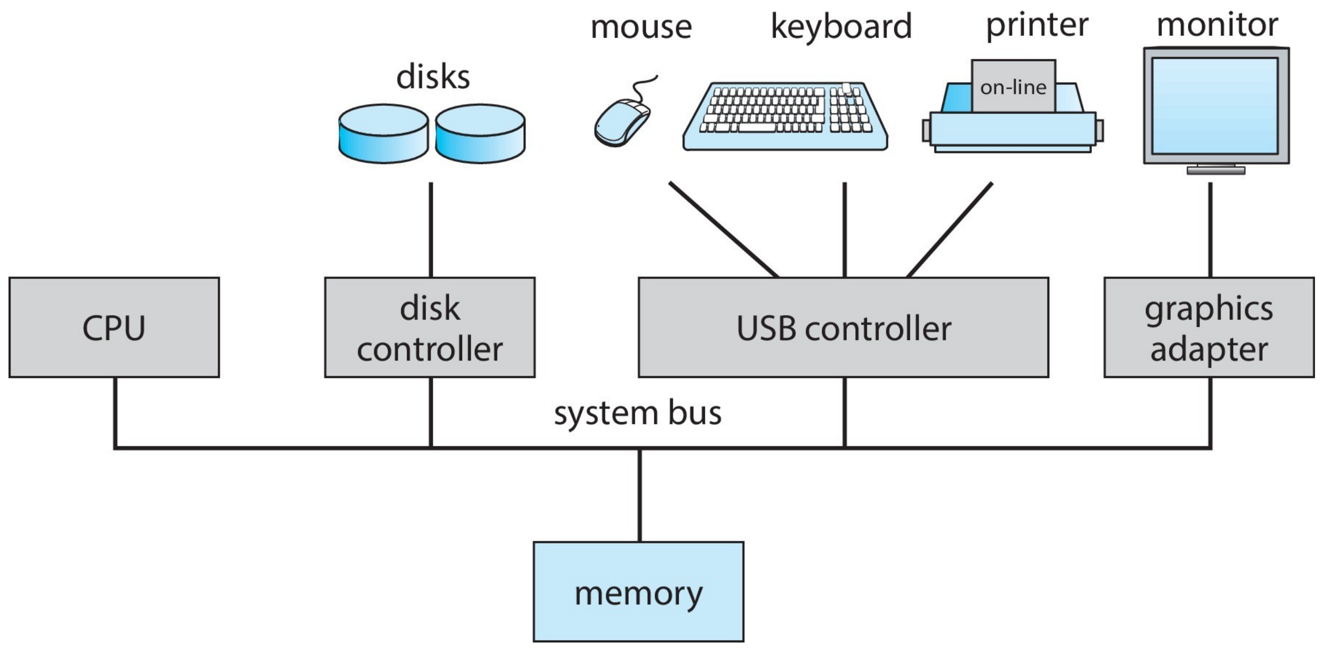
2. Middleware frameworks that ease application development and provide features, and

3. System programs that aid in managing the system while it is running.

1.2 Computer System Organization

Computer System Organization

- A general-purpose computer system consists of one or more **CPUs** and a number of device controllers connected through a common **bus** that provides access between components and shared memory
  - Typically, OS have a device driver for each device controller
  - The CPU and device controllers can execute in parallel, competing for memory cycles



Interrupts

Interrupts are a key part of how OS and hardware interact.

- The occurrence of an event is usually signaled by an **interrupt** from either the hardware or the software. Interrupts alert the CPU to events that require attention.
- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to interrupt service routine. On completion of the interrupt service routine, the CPU resumes the interrupted computation.
- The **interrupt vector** contains the addresses of all the service routines. Only a predefined number of interrupts is possible.
- The interrupt architecture must also save the state information of whatever was interrupted, so that it can restore this information after servicing the interrupt.

Interrupts (cont.)

- Most CPUs have two interrupt request lines.:
  - Nonmaskable interrupts – for events like unrecoverable memory errors
  - Maskable interrupts – can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted.
    - maskable interrupt is used by device controllers to request service
- Interrupts are used throughout modern operating systems to handle asynchronous events
- Device controllers and hardware faults raise interrupts.
- To enable the most urgent work to be done first, modern computers use a system of interrupt priorities.

Storage

- All forms of **memory** provide an **array of bytes**.
- Each byte has its own address.

Storage Definitions and Notation

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.
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.

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 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).

Memory

The **CPU** can load instructions only from **memory**

- So any programs must first be loaded into memory to run.
- General-purpose computers run most of their programs from rewritable memory, called **main memory** (also called random-access memory, or RAM).
- Main memory commonly is implemented in a semiconductor technology called **dynamic random-access memory (DRAM).**

Instruction–execution cycle

As executed on a system with a **von Neumann architecture**,

1. first **fetch**es an instruction **from memory** and stores that instruction in the **instruction register**.
2. The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register.
3. After the instruction on the operands has been executed, the result may be **store**d back **in memory**.

Main Memory

Ideally, we want the programs and data to reside in main memory permanently, but:

1. Main memory is usually too small to store all needed programs and data permanently.
2. Main memory is a **volatile** storage device that loses its contents when power is turned off or otherwise lost.

Secondary storage

Large **nonvolatile** storage capacity.

Most programs (system and application) are stored in secondary storage until they are loaded into memory.

Many programs then use secondary storage as both the source and the destination of their processing

1. **Non-volatile memory (NVM)** devices– faster than hard disks. **electrical semiconductor**-based electronic circuits
   - non-volatile memory chips (Flash memory Storage) – EEPROM, SSD, etc.
   - Becoming more popular as capacity and performance increases, price drops
2. **Hard Disk Drives** (**HDDs**) – 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

Tertiary storage

- CD-ROM, blue-ray - optical
- magnetic tapes, and so on

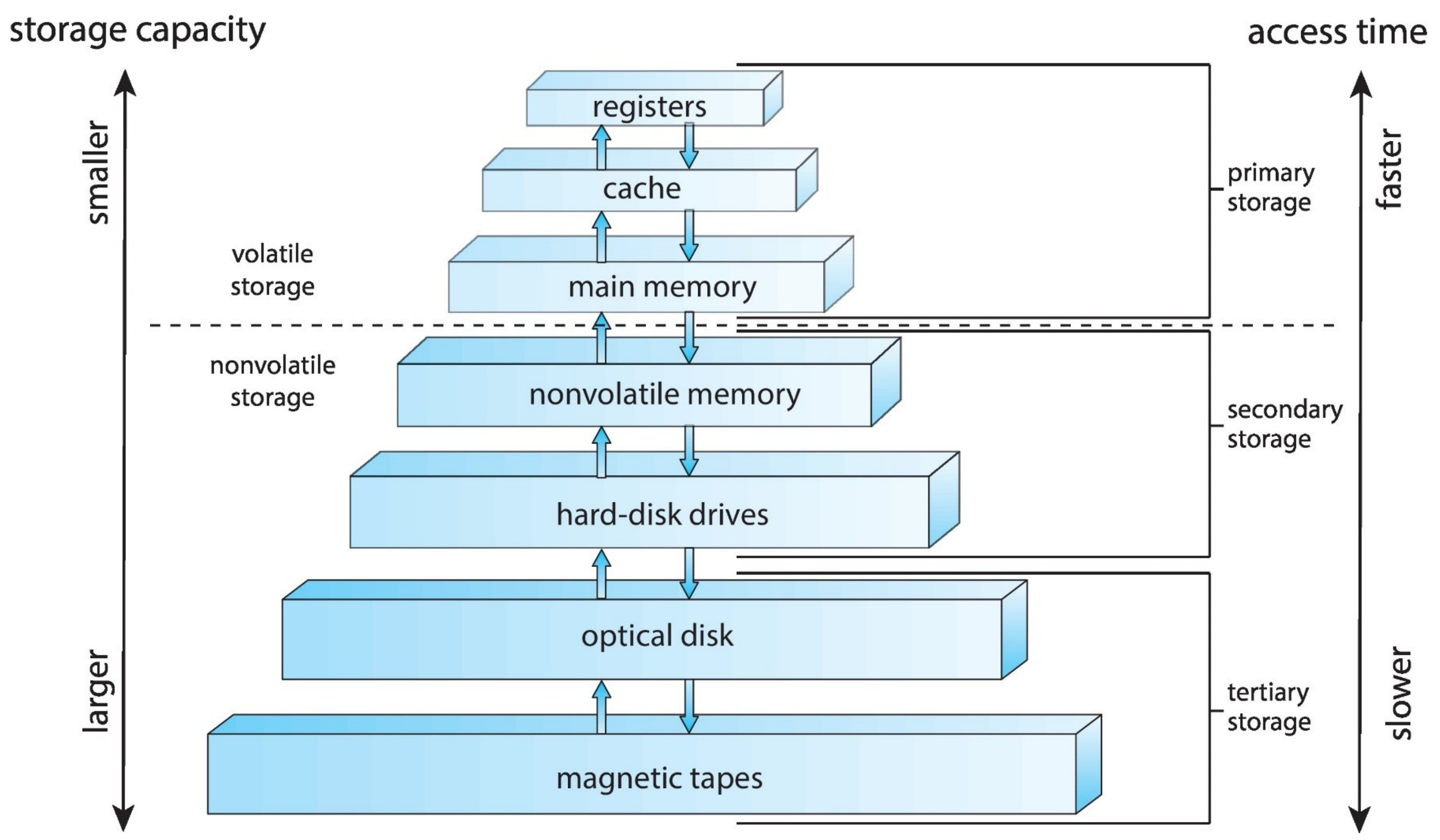
nonvolatile
Slow and large.
For special purposes like storing backup copies

Storage Hierarchy

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

- **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



1.3 Computer System Architecture

Computer-System Architecture

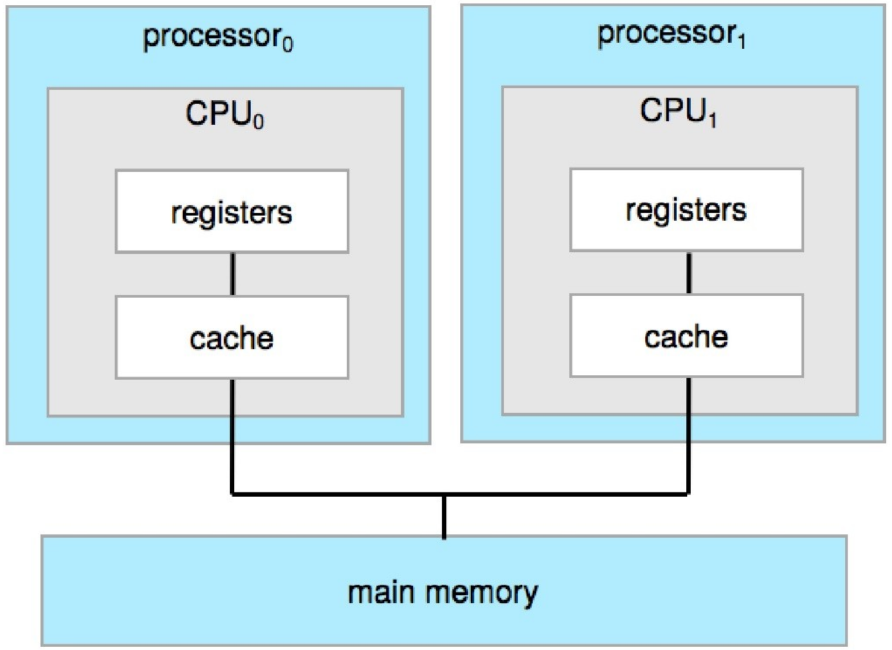
Modern **Multiprocessor systems**,

Systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

- from mobile devices to servers, have two (or more) processors, each with a single-core CPU
- also known as **parallel systems**, **tightly-coupled systems**
- advantages include increased throughput, economy of scale, increased reliability – graceful degradation or fault tolerance

Two types:

1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
2. **SMP - Symmetric Multiprocessing** – each peer processor performs all tasks, including OS functions and user processes.
   - Most common.

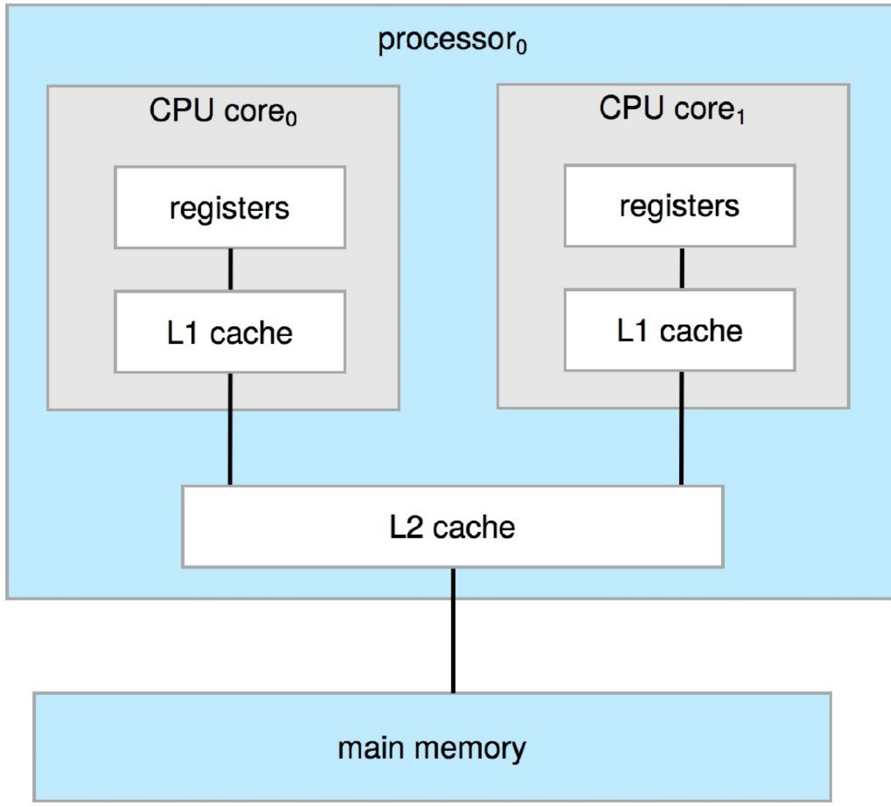


Symmetric Multiprocessing Architecture

Multicore Systems

Multicore

- Multiple computing cores reside on a single chip
- More efficient than Multi-chips with single core
- On-chip communication faster than between-chip communication
- Use significantly less power - important for mobile devices as well as laptops



Level 2 cache in the Duel-Core design is local to the chip, is smaller, and is shared by processing cores

A Dual-Core Design with two cores on the same processor chip

Multicore SMP Systems

- Windows
- macOS
- Linux
- Android
- iOS

Virtually all modern OS

1.4 Operating System Operations

Computer Startup

To start running, a computer needs an initial program, or **bootstrap** program

- Typically, it is stored within the computer hardware in **read-only memory (ROM)** or electrically erasable programmable read-only memory (**EEPROM**), known by the general term **firmware** (ie., storage that is nonvolatile, and cannot be written to frequently)

- Initializes all aspects of the system, from CPU registers to device controllers to memory contents.

- Bootstrap program locates **operating-system kernel and loads into memory**

Computer Startup (cont.)

Once the **kernel** is loaded and executing, it can start providing services to the system and its users.

Some services are provided outside of the kernel by system programs that are loaded into memory at boot time to become **system daemons**, which run the entire time the kernel is running.

On Linux, the first system program is “systemd,” and it starts many other daemons.

Once this phase is complete, the system is fully booted, and the system waits for some event to occur.

Interrupt-driven OS

If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen.

Events are almost always **signal**ed by the occurrence of an interrupt.

A **trap** (or an **exception**) is a software-generated interrupt caused by:

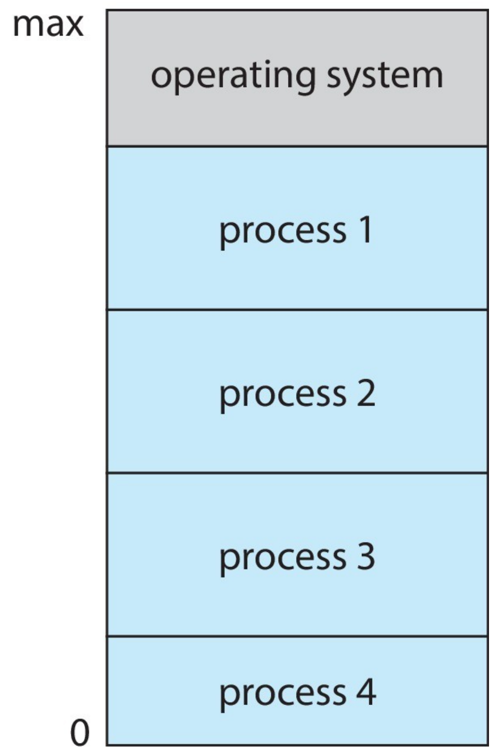
1. An error (for example, division by zero or invalid memory access)
2. A specific request from a user program that an operating-system service be performed by executing a special operation called a **system call**.

Multiprogramming

One of the most important aspects of operating systems is the ability to run multiple programs, as a single program cannot, in general, keep either the CPU or the I/O devices busy at all times.

- **Multiprogramming** increases CPU utilization, as well as keeping users satisfied, by organizing programs so that the CPU always has one to execute.
- The operating system keeps several processes in memory simultaneously.
- The operating system picks and begins to execute one of these processes. Eventually, the process may have to wait for some task, such as an I/O operation, to complete.
- In a non-multiprogrammed system, the CPU would sit idle.
- In a multiprogrammed system, the operating system simply switches to, and executes, another process.
- When ***that*** process needs to wait, the CPU switches to ***another*** process, and so on.
- Eventually, the first process finishes waiting and gets the CPU back.
- As long as at least one process needs to execute, the CPU is never idle.

Memory Layout for Multiprogrammed System



Multitasking

Multitasking is a logical extension of multiprogramming.

- In multitasking systems, the CPU executes **multiple processes by switching among them**, but the switches occur frequently, providing the user with a fast **response time**.

- Consider that when a process executes, it typically executes for only a short time before it either finishes or needs to perform I/O.

- I/O may be interactive; that is, output goes to a display for the user, and input comes from a user keyboard, mouse, or touch screen.

- Since interactive I/O typically runs at “people speeds,” it may take a long time to complete. Rather than let the CPU sit idle as this interactive input takes place, the operating system will rapidly switch the CPU to another process.

Dual-mode and Multi-mode Operation

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

- **User Mode** and **Kernel Mode**
  - System **boot time** hardware starts in **kernel mode**
  - OS **is then** loaded and starts user applications in user mode

- Increasingly CPUs support **Multi-mode** operations
  - i.e. **virtual machine manager** (**VMM**) **mode** for guest **VMs** – more power than user processes, but fewer than kernel
  - Intel processors have 4 modes , 0 (kernel), 1 &2 (OS services), 3 (user)

Dual-mode

Two separate ***modes*** of operation:

1. **user mode** and
2. **kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**)

A bit, called the **mode bit**, is added to the **hardware** of the computer to indicate the current mode: kernel (0) or user (1).

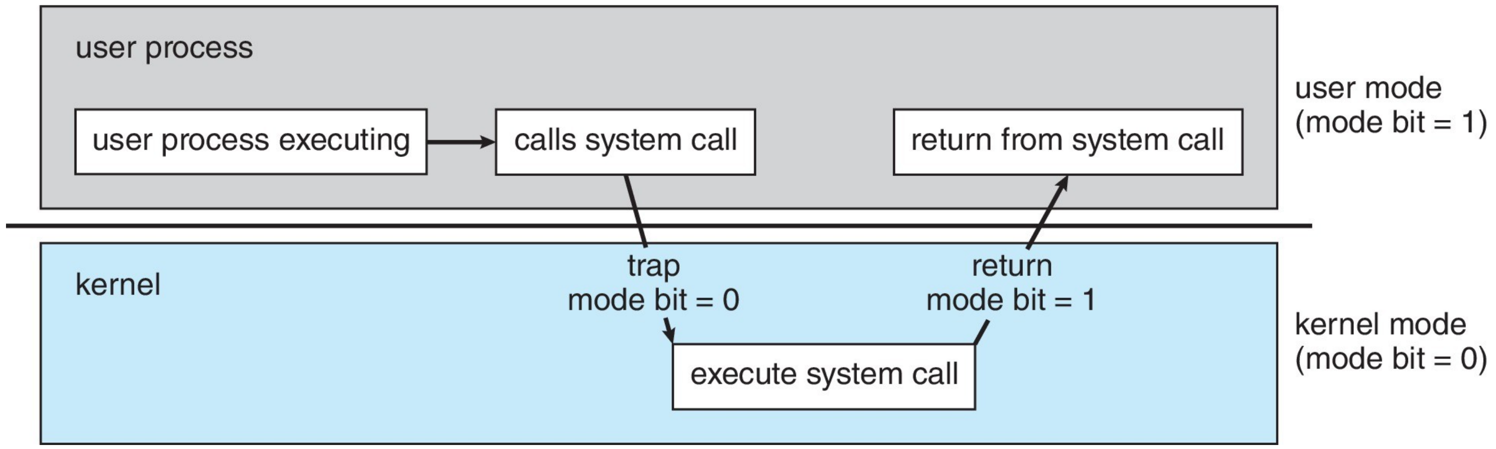
Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0).

Thus, whenever the operating system gains control of the computer, it is in kernel mode.

The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program

Transition from User to Kernel Mode

- **Mode bit** provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - System call changes mode to kernel, return from call resets it to user



1.5 Resource Management

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, etc.
  - Initialization data
- **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

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

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

File-system Management

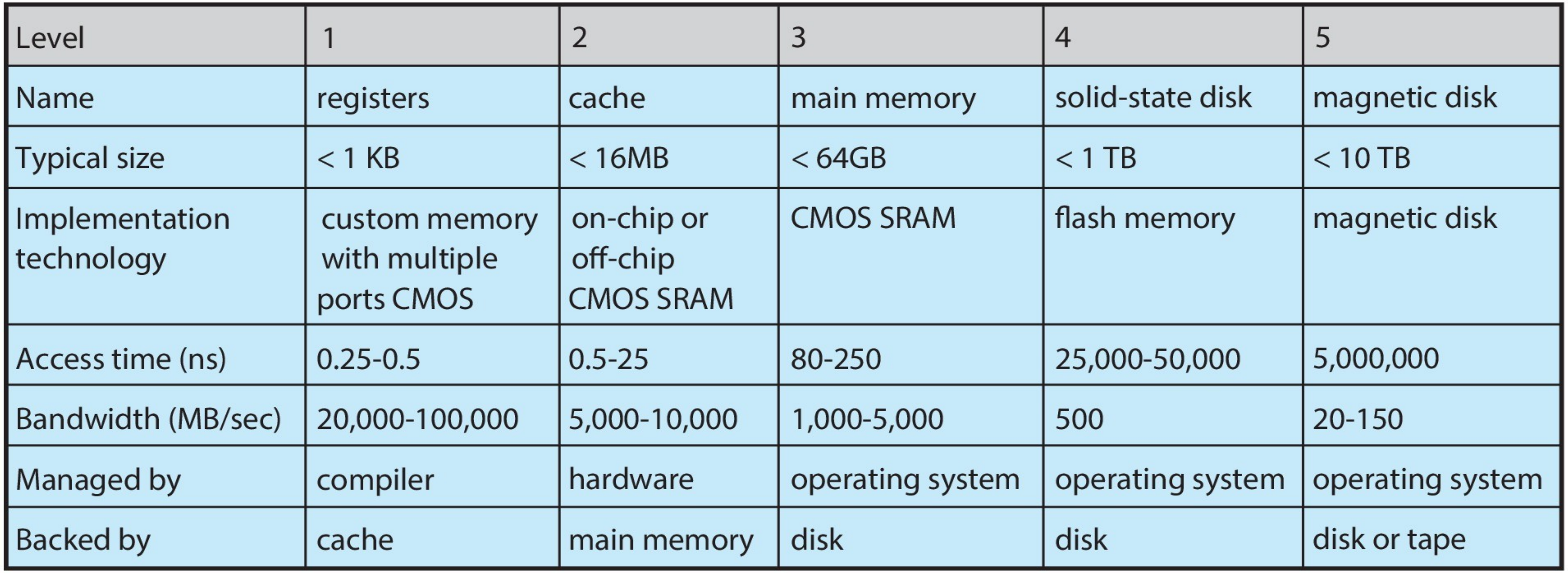
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by **device** (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - **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

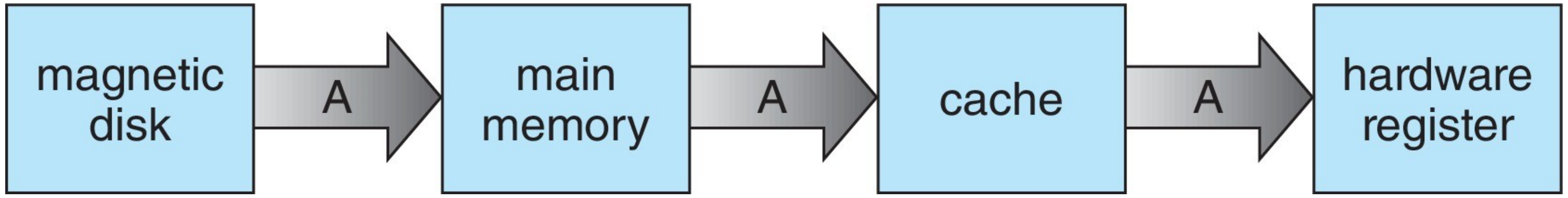
- Usually **disks** used to **store** data that does not fit in main memory or data that must be kept for a “long” **period of time**
- Proper management is of central importance
- Entire **speed of computer operation** hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications

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

Migration of data “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

Protection and Security

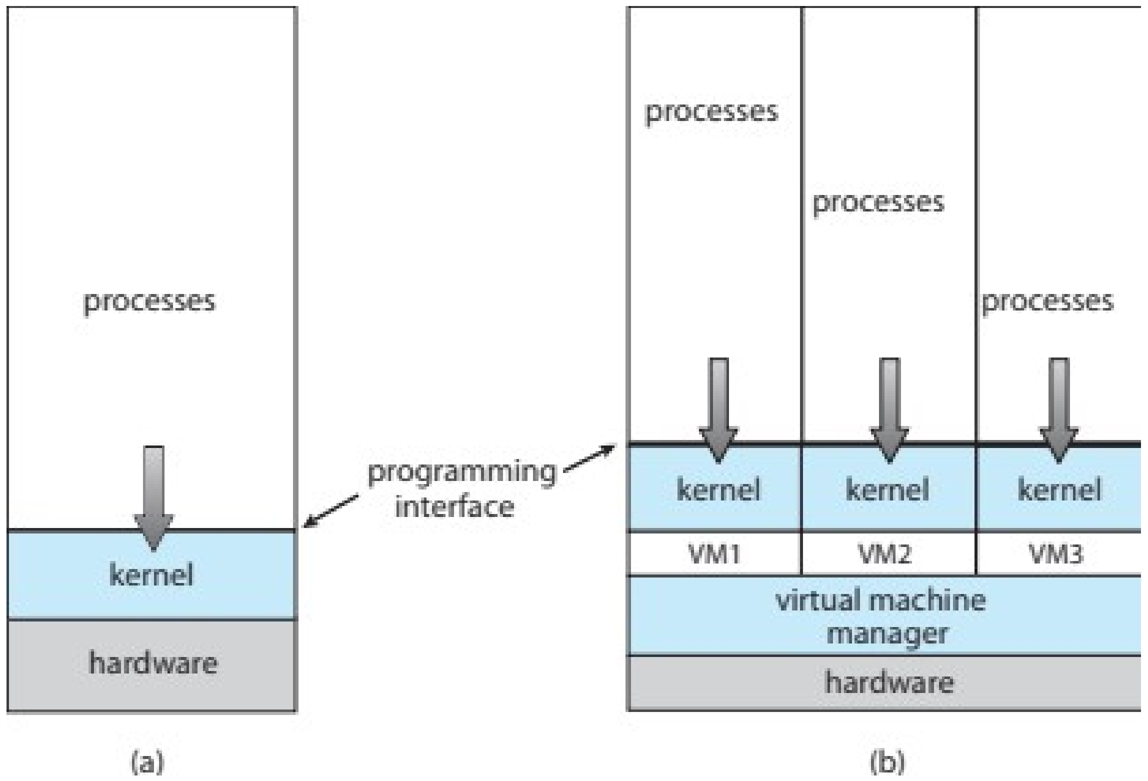
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - 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** (extra permissions given by OS) allows user to change to effective ID with more rights

Virtualization

- **Hardware virtualization** or *platform virtualization* refers to the **creation of a virtual machine that acts like a real computer with an operating system**. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Arch Linux may host a virtual machine that looks like a computer with the Microsoft Windows operating system; Windows-based software can be run on the virtual machine.
- **Emulation** involves **simulating computer hardware in software**, typically used when the source CPU type is different from the target CPU type. For example, when Apple switched from the IBM Power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulation facility called “Rosetta,” which allowed applications compiled for the IBM CPU to run on the Intel CPU.
  - That same concept can be extended to allow an entire operating system written for one platform to run on another.

Computing Environments - Virtualization

- **Virtualization** allows operating systems to **run as applications within other operating systems**.
- VMware created a new virtualization technology in the form of an application that ran on Windows. That application ran one or more **guest** copies of Windows or other native x86 operating systems, each running its own applications.
- Windows was the **host** operating system, and the VMware application was the **virtual machine manager (VMM)**. The VMM runs the guest operating systems, manages their resource use, and protects each guest from the others.



Distributed Systems

A **distributed system** is a **collection of physically separate**, possibly heterogeneous computer systems that are **network**ed to provide users with access to the various resources that the system maintains.

- Access to a **shared resource increases computation speed, functionality, data availability, and reliability**.
- **TCP/IP** most common network protocol, and it provides the fundamental architecture of the **Internet**
- Networks are characterized based on the distances between their nodes:
    - **Local Area Network** (**LAN**)
    - **Wide Area Network** (**WAN**)
    - **Metropolitan Area Network** (**MAN**)
    - **Personal Area Network** (**PAN**)
- **Network Operating System** provides features between systems across network
    - Communication scheme allows systems to exchange messages
    - Illusion of a single system

Free (free/libre) and Open-Source Operating Systems

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

- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**
  - Copyleft - software work may be used, modified, and distributed freely on condition that anything derived from it is bound by the same condition
  - Free software and open-source software are two different ideas championed by different groups of people
    - http://gnu.org/philosophy/open-source-misses-the-point.html/
- Examples include **GNU/Linux** and **BSD UNIX** (including **Darwin**, the core kernel of **macOS**), and many more
- Free Virtualbox VMM tool (http://www.virtualbox.org)
  - Use to run guest operating systems for exploration

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) providesa 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.