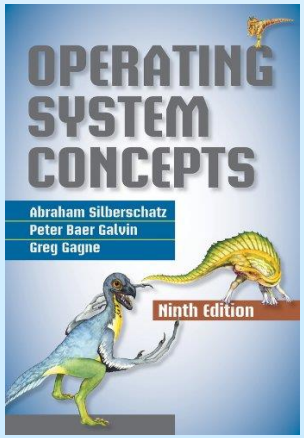
Operating Systems

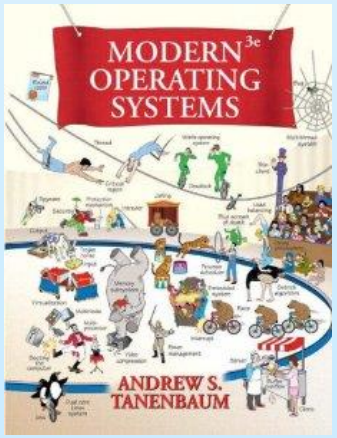
31261

Spring 2013, Ort Braude College
Electrical Engineering Department

Bibliography

- Silberschatz and Galvin. Operating Systems Concepts. 8th edition, 2008, John Wiley & Sons, Inc.
- Andrew S. Tanenbaum. Modern Operating Systems, 3/e. Prentice-Hall 2007
- Bovet and Cesati. Understanding the Linux Kernel. 3rd edition, 2005, O'Reilly.
- Robert Love. Linux Kernel Development. Third Edition , 2010, Addison-Wesley Professional.
- Python course site: http://samyzaf.com/braude/PYTHON





Course Web Site

- **Courses Web Site:**

  http://www.samyzaf.com/braude

- **Slides based on two books complimentary materials:**

  1. Silberschatz, Galvin and Gagne, *Operating System Concepts Presentation Slides*
  2. Andrew S. Tanenbaum, *Modern Operating Systems*, 3/e. Prentice-Hall 2007

Software

- Python software should be downloaded from

  http://www.samyzaf.com/braude/PYTHON

  Into a personal flash drive (diskonkey)
  - at least 2GB drive is needed
- All software can be executed from the flash drive on any standard Windows PC
- So you can do your coding work at home and everywhere you have an access to a windows PC
- We will also use a few Linux sessions in College Linux labs, or may connect to a Linux server from a windows client program (such as putty.exe). Details will follow later.

Chapter 1: Introduction

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

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization

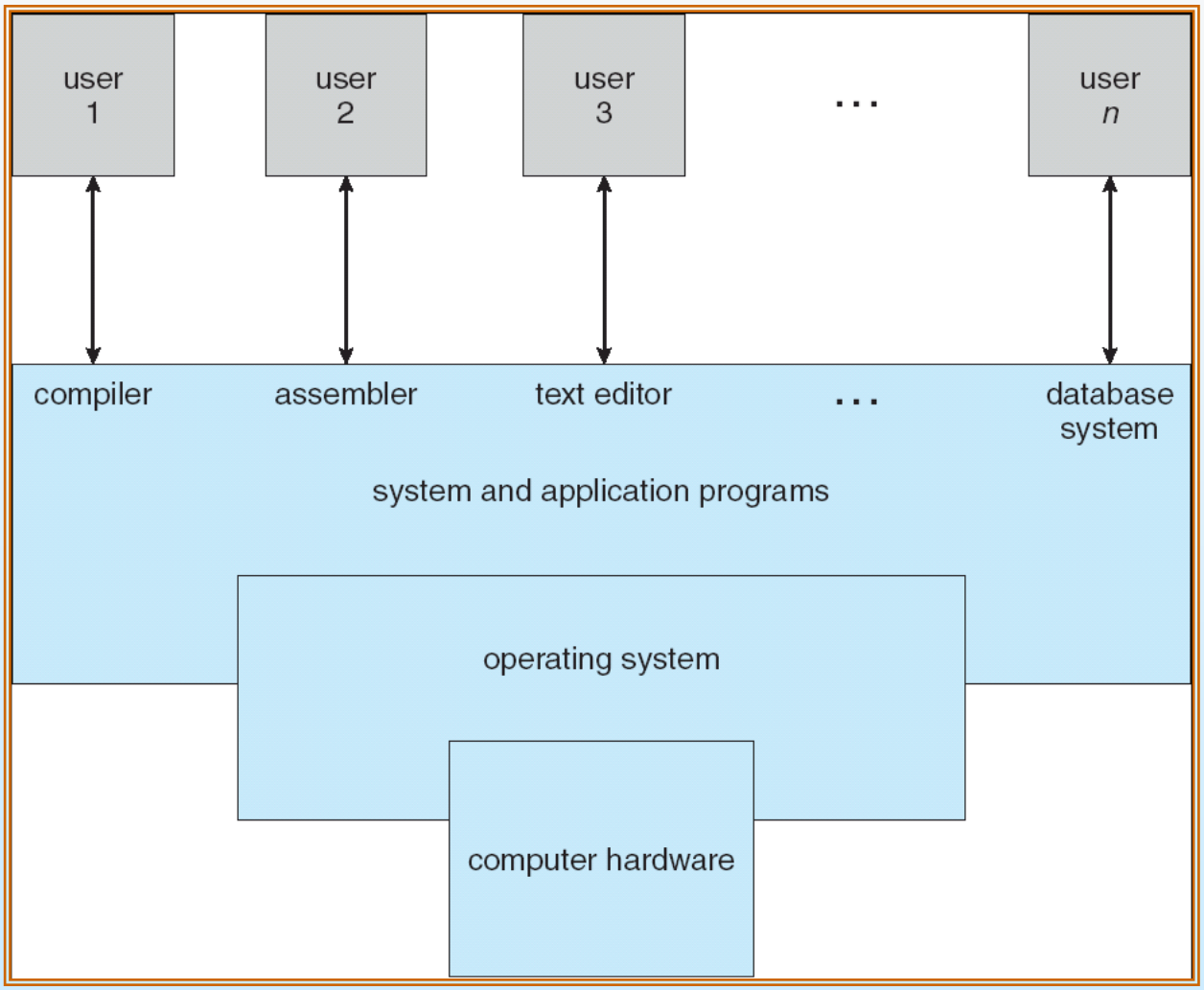
What is an Operating System?

- 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, devices

Four Components of a Computer System



The First Computer: ENIAC 1946

Electronic Numerical Integrator And Computer

Electrical Engineering at 1946

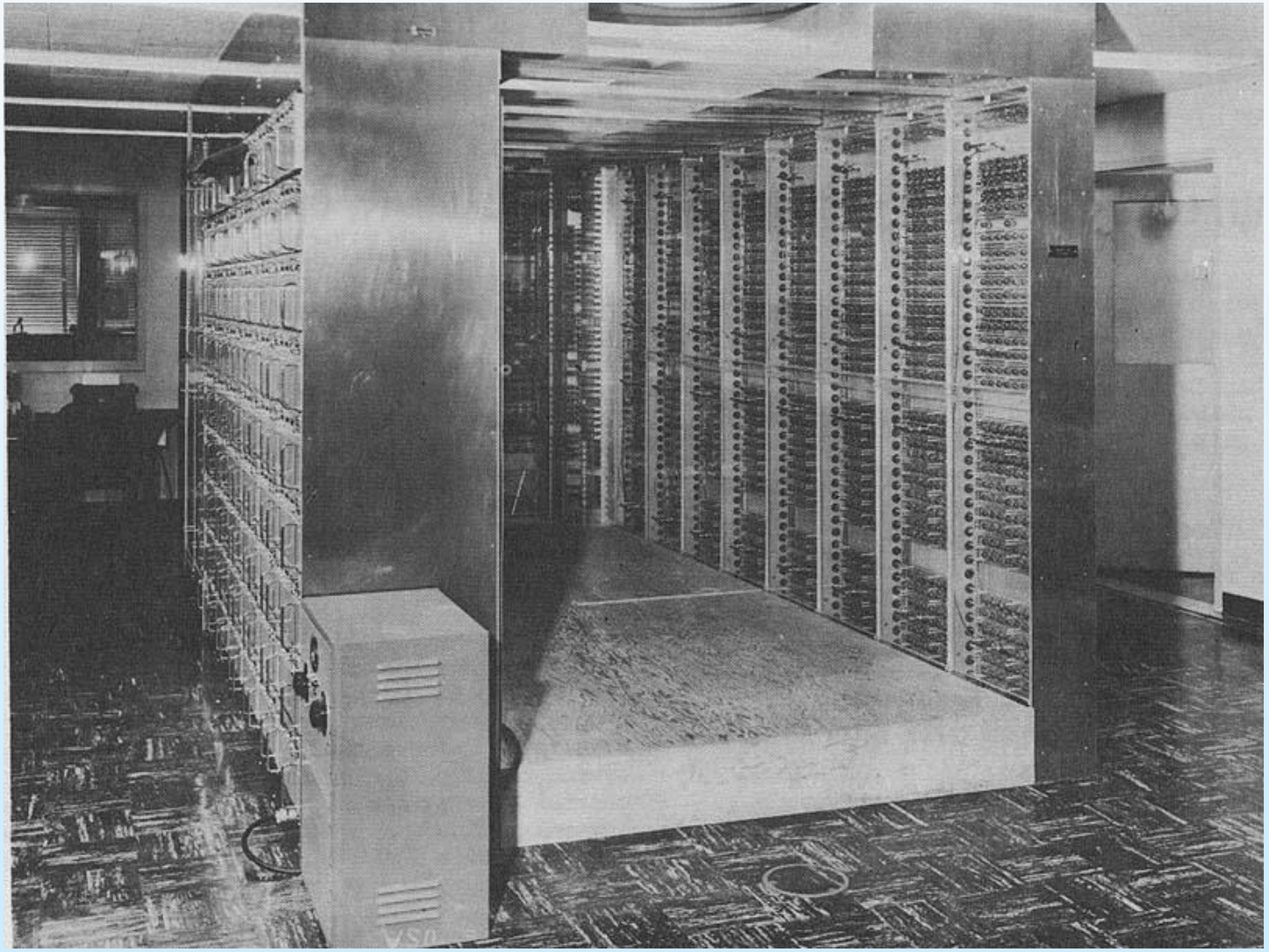
"המחשבים בעתיד ישקלו <u>לא יותר מ-1.5 טון</u>"

Popular Mechanics חוזים את התקדמות המדע, 1949

משקל - 27,215 ק"ג
מכיל יותר מ-18,000 שפופרות קטודיות
מדי חודש הוחלפו 2,000 מהן על-ידי 6 טכנאים
מחיר – כ-750,000 $ (של אותם ימים)
שטח – 167 מ"ר. שימש לתכנון פצצת המימן הראשונה

Second Computer: MANIAC I, 1952

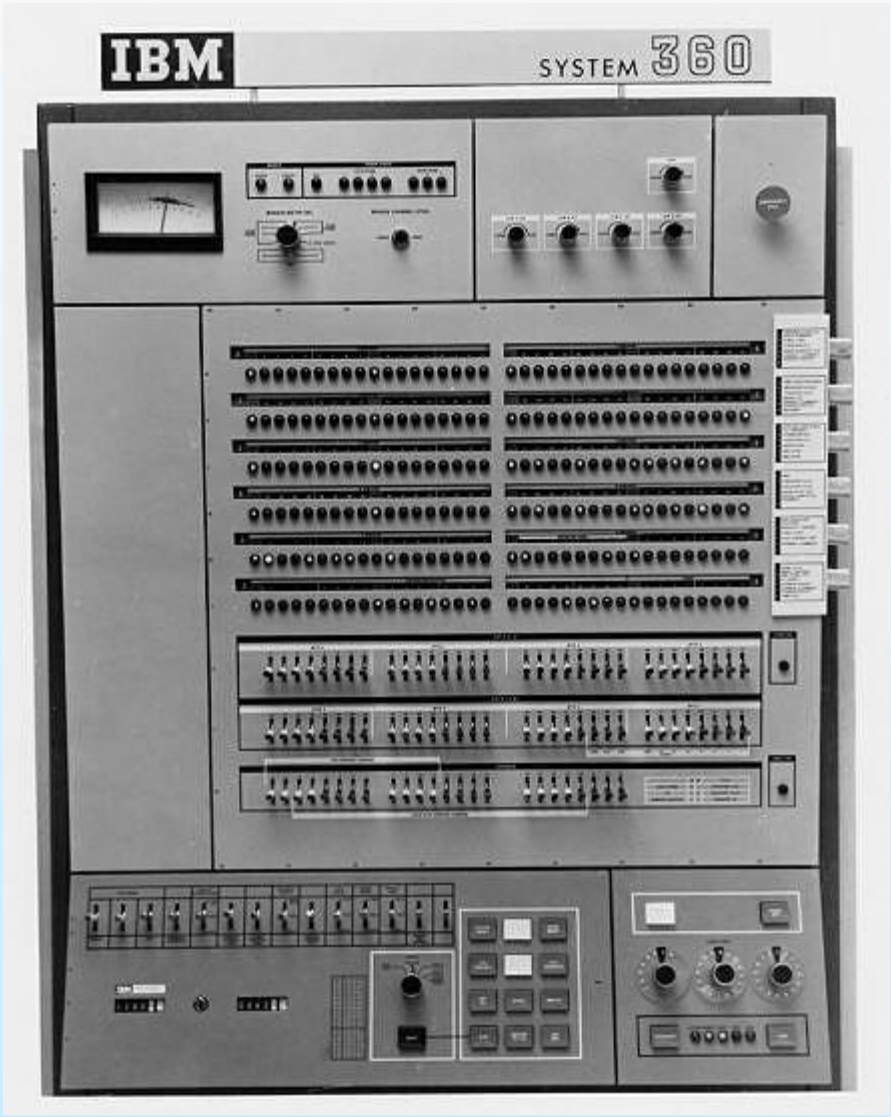
***M**athematical **A**nalyzer, **N**umerical **I**ntegrator, **a**nd **C**omputer*



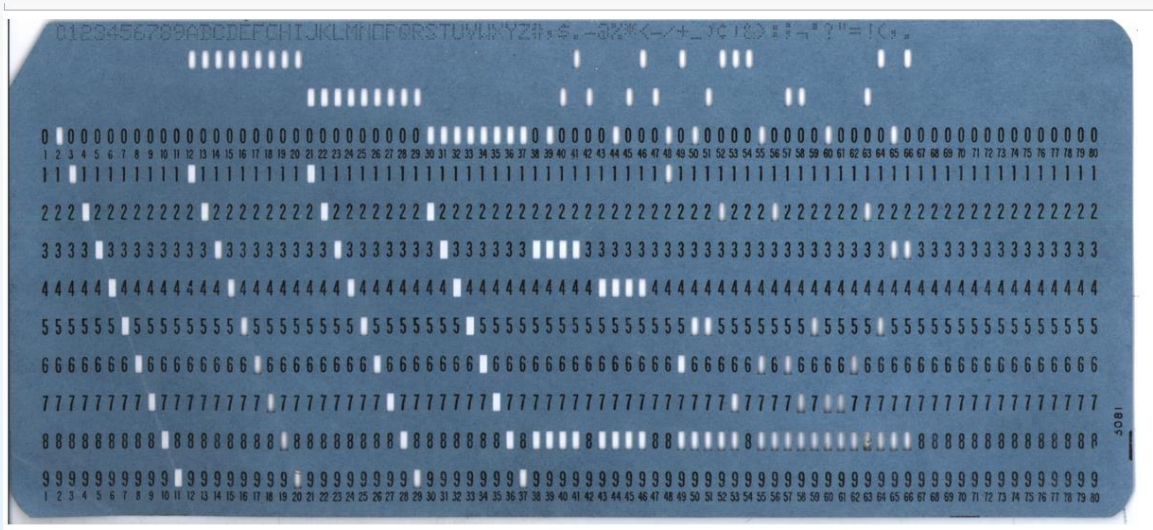
IBM Model 701 (Early 1950's)

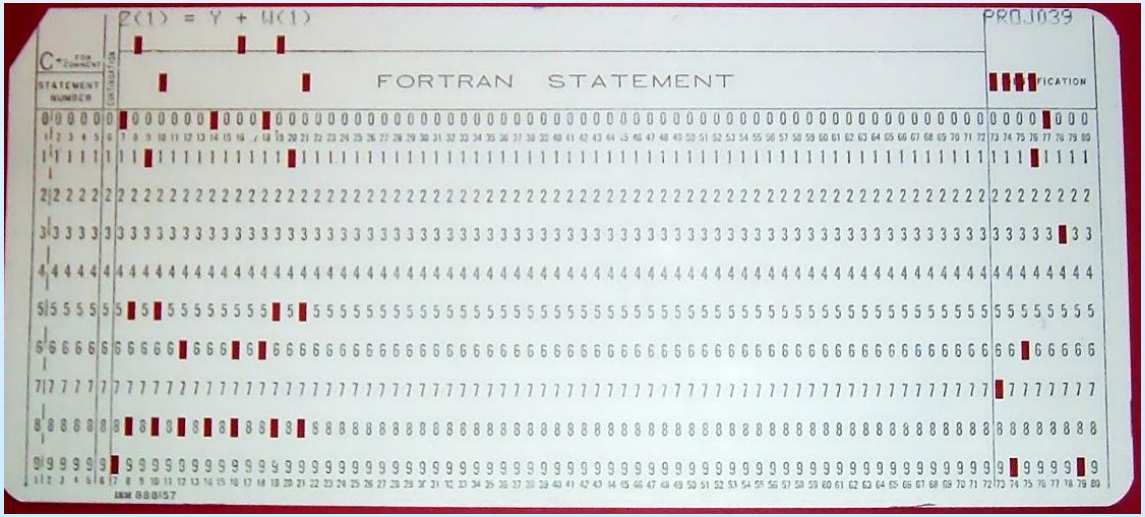
IBM 7094 (Early 1960's)

IBM System 360 Console



Punch Cards





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.)

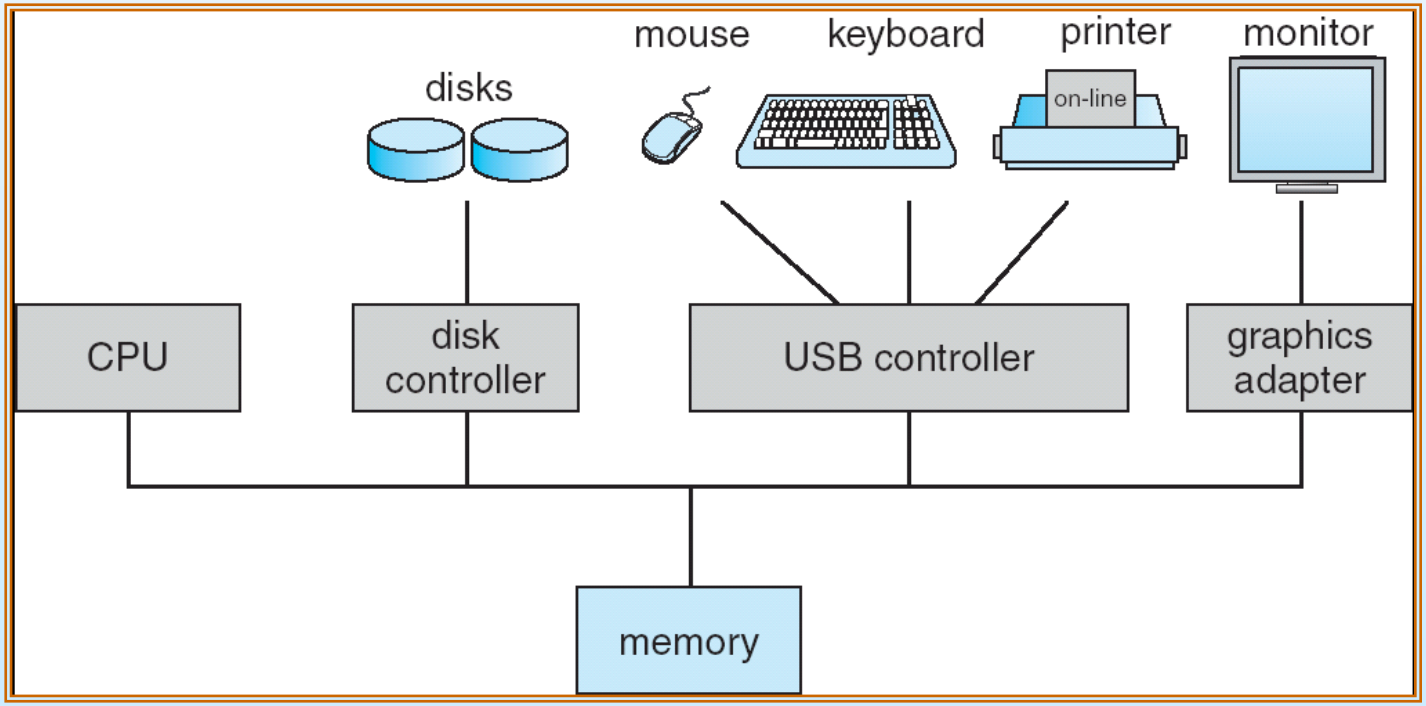
- No universally accepted definition
- ***"Everything a vendor ships when you order an operating system"*** is a good approximation
  - But varies wildly
- "***The one program running at all times on the computer***" is the **kernel**
- Everything else is either a
  - **System program**
    ships with the operating system
  - **Application program**
    Written by a developer on top of the operating system

Computer Startup

- **Bootstrap Program**
  - Loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution
- **EEPROM**
  Electrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory

Computer System Organization

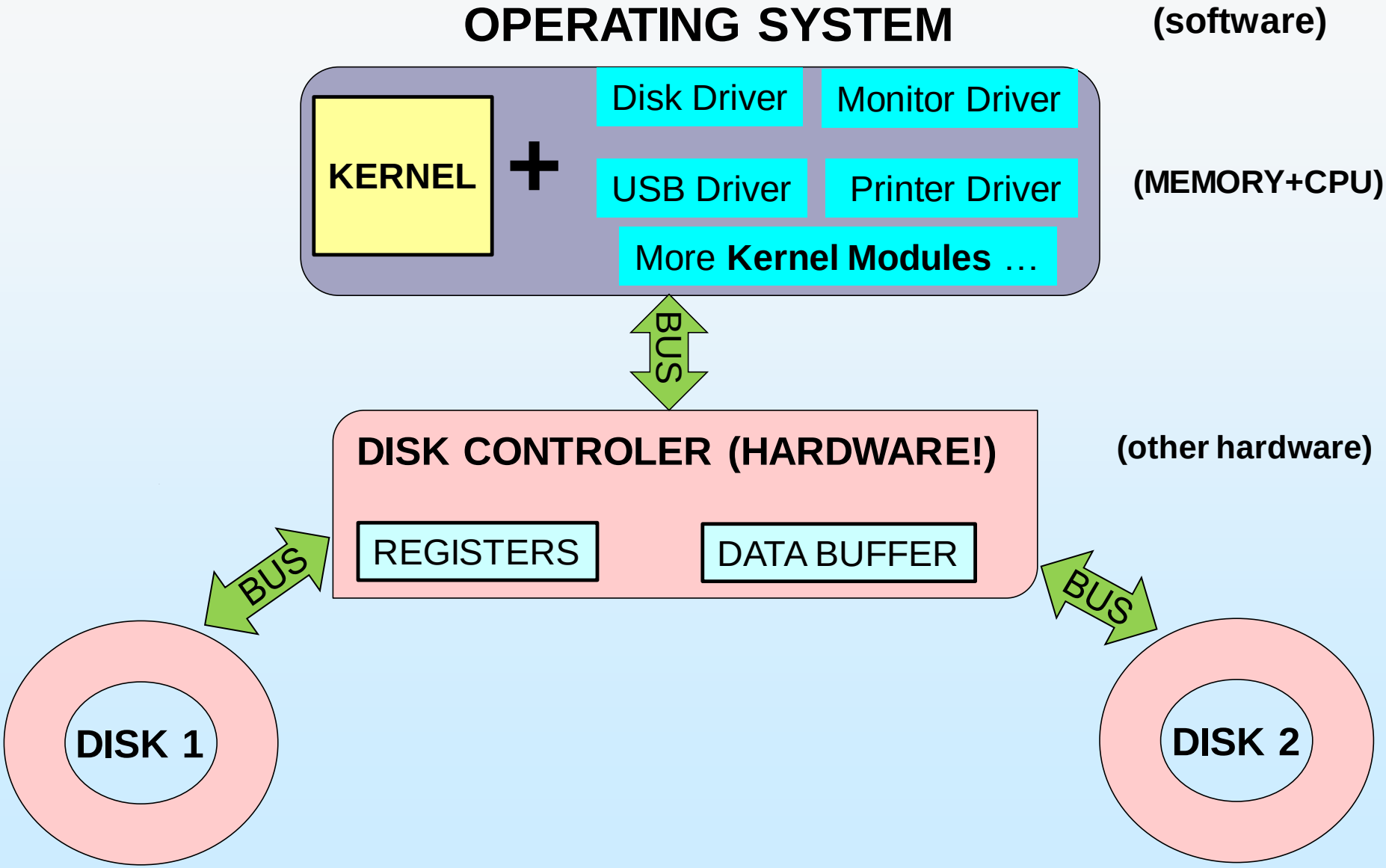
- Computer-system operation
  - One or more CPUs (cores), device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



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*

OS ➔ DRIVER ➔ CONTROLER ➔ DEVICE



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.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.

Hardware and Software Interrupts

- **Hardware interrupt**
  - sent to the processor from another device, like a disk controller , keyboard, or an external peripheral
  - Example: pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position
  - Hardware interrupt is referred to as an interrupt request (IRQ)
- **Software interrupt**
  - Caused either by an exceptional condition in the processor itself or by a special instruction in the instruction set which requests an interrupt when it is executed
  - AKA a *trap* or *exception* and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself
  - Example: if the arithmetic logic unit will generate a *divide by zero exception* if it is requested to divide a number by zero

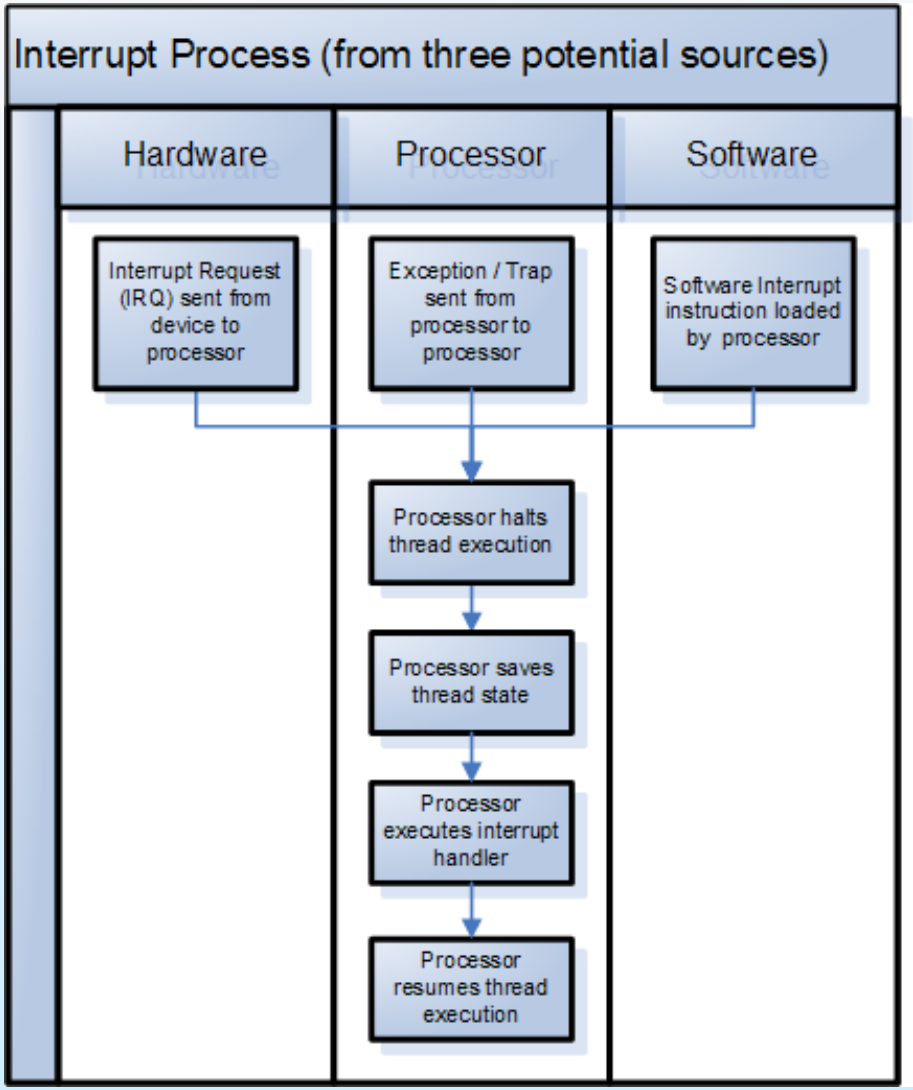


Figure: courtesy of the Wikimedia Foundation

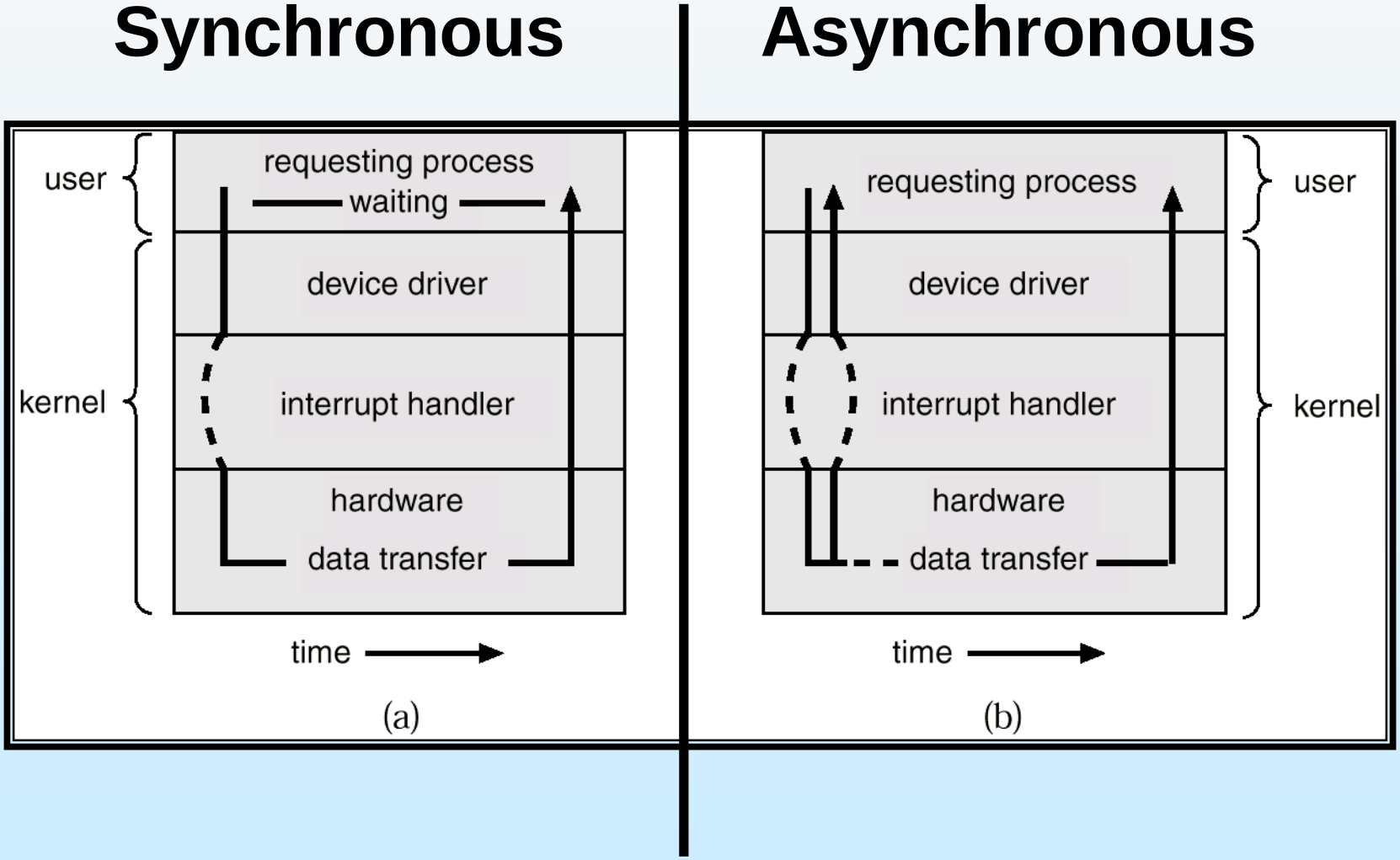
Interrupt Handling

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

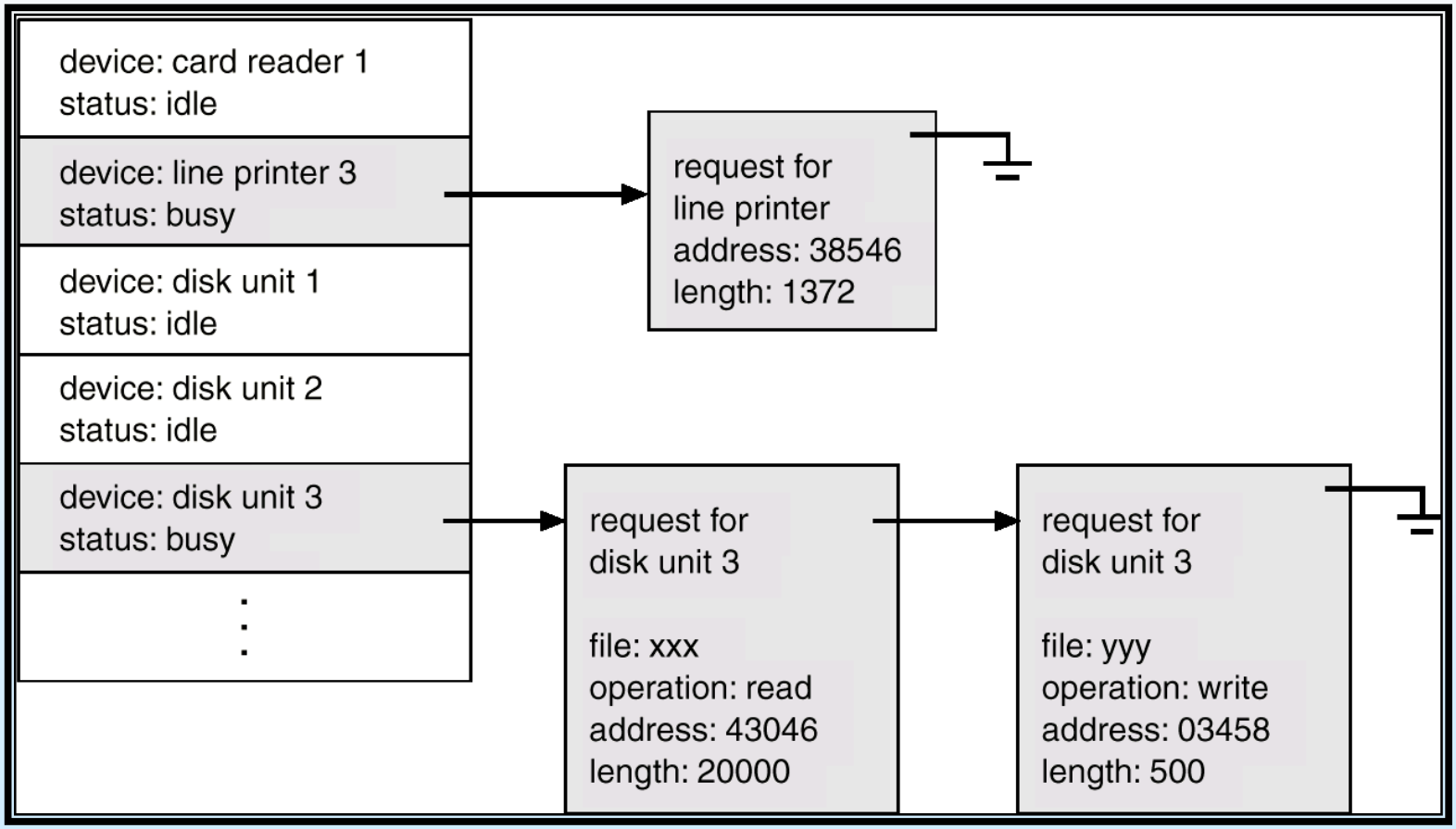
I/O Structure

- In synchronous mode: after I/O starts, user program goes to “sleep”
- CPU control will be given to other program which is ready for action
- 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.
- ***System call***
  request to the operating system to allow user to wait for I/O completion.
- ***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



Device-Status Table

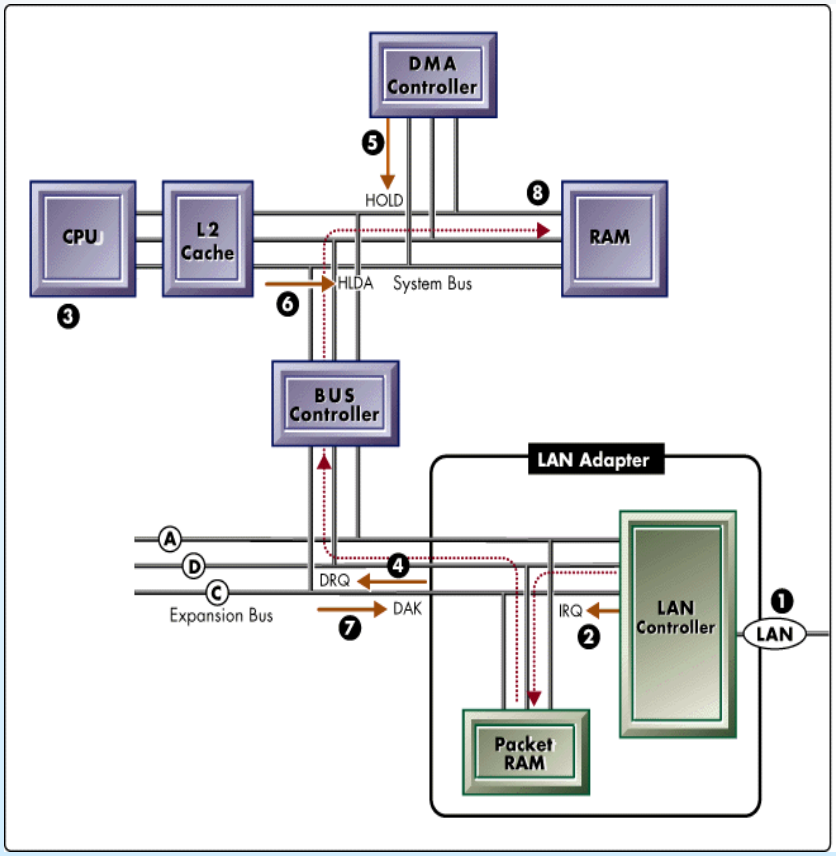


Direct Memory Access Structure

- 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.

DMA – Direct Memory Access

- The CPU is too expensive to be engaged with slow I/O transfers:
  - A typical CPU operates at several GHz (i.e., several 10^9 instructions per second)
  - A typical hard disk has a rotational speed of 7200 revolutions per minute for a half-track rotation time of 4 ms
  - This is 4 million times slower than the processor!
- Instead the CPU initiates a transfer with the DMA, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done
- So in effect, the DMA is a mini-controller that works for the CPU and does I/O transfer jobs for it
- Some systems contain several DMA's
- DMA is also used for "memory to memory" copying in multi-core processors
- Hardware systems such as disk drives, graphic network and sound cards use the DMA for passing data around



http://support.novell.com/techcenter/articles/ana19950501.html

Storage Structure

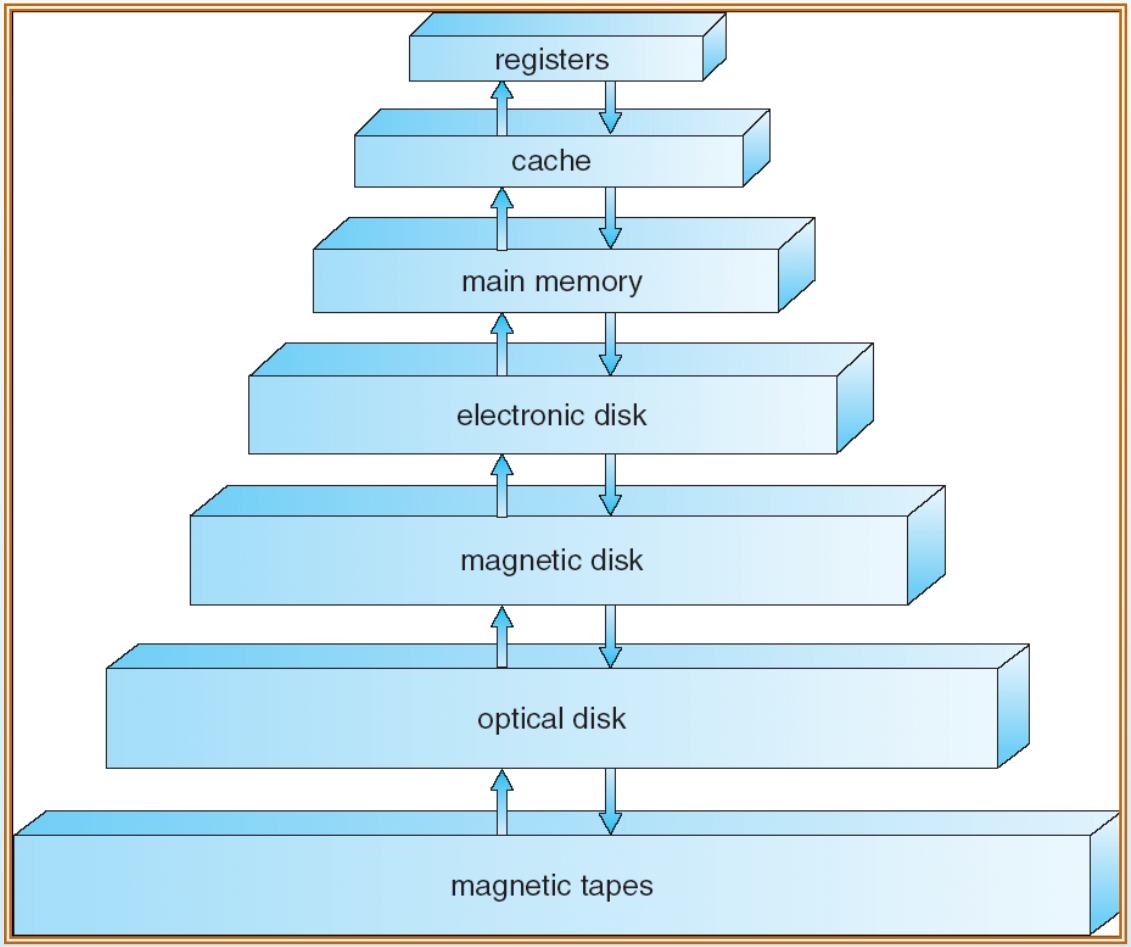
- **Main memory**
  The only large storage media that the CPU can **access directly**
- **Secondary storage**
  Extension of main memory that provides large nonvolatile storage capacity
- **Magnetic disks**
  Rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer

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 (closest cache in modern CPU's: … L1, L2, L3)

Storage-Device Hierarchy



Approximate timing for various operations on a typical PC

execute typical instruction	1/1,000,000,000 sec = 1 nanosec
fetch from L1 cache memory	0.5 nanosec
branch misprediction	5 nanosec
fetch from L2 cache memory	7 nanosec
Mutex lock/unlock	25 nanosec
fetch from main memory	100 nanosec
send 2K bytes over 1Gbps network	20,000 nanosec
read 1MB sequentially from memory	250,000 nanosec
fetch from new disk location (seek)	8,000,000 nanosec
read 1MB sequentially from disk	20,000,000 nanosec
send packet US to Europe and back	150 milliseconds = 150,000,000 nanosec

Source: http://norvig.com/21-days.html#answers

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 important design problem
  - Cache size and replacement policy

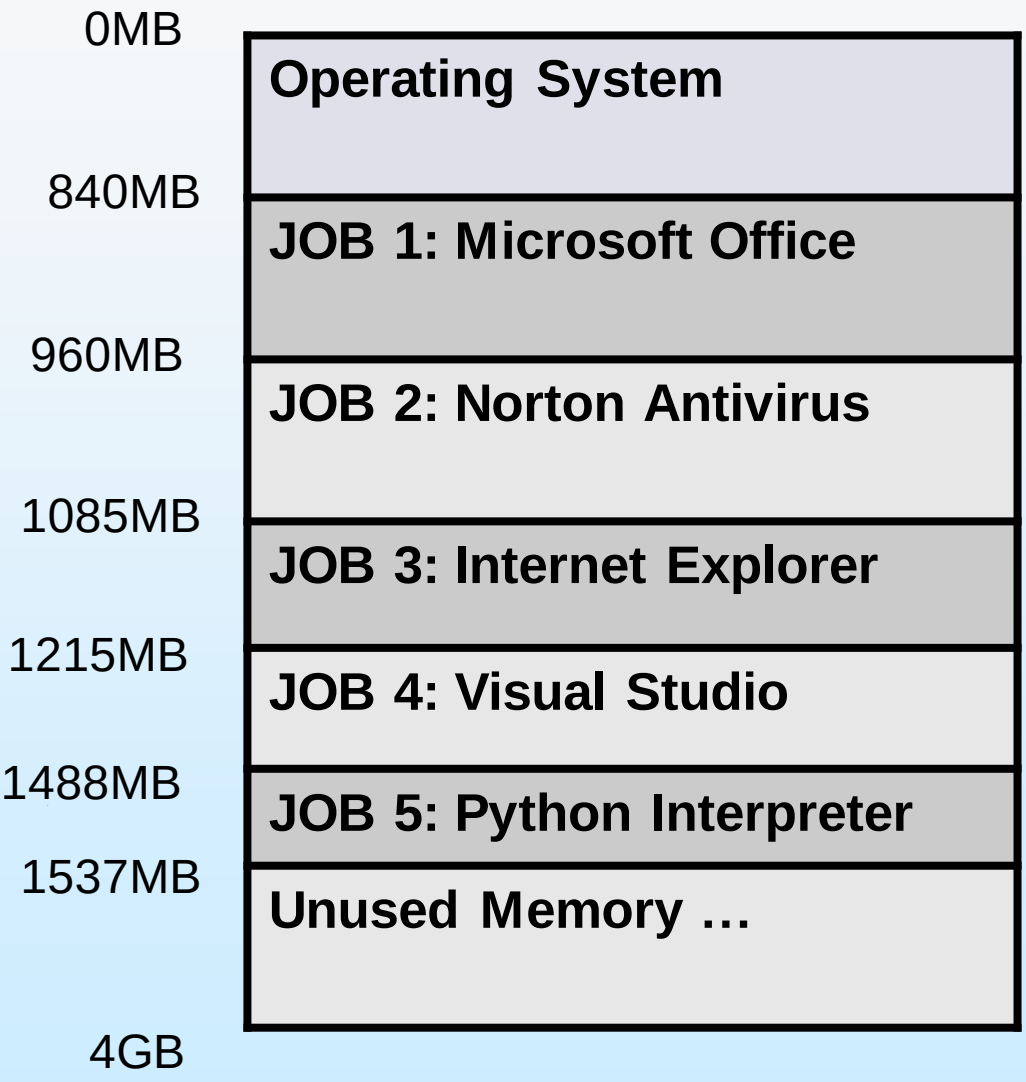
Multi-Programming

- **Multiprogramming** 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

Timesharing (Multitasking)

- **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



Operating-System Operations

- Interrupts are a critical mechanism for operating systems!
- Interrupts are generated **hardware** and **software**
  - **Hardware Interrupts**
  - **Software Interrupts**
- Software interrupts are caused by **exception** or **trap**
  - **Division by zero**
  - **Integer or Float Overflow**
  - Request for operating system services:
    - **read**
    - **write**
    - **exit**
- Other process problems include infinite loop, processes modifying each other or the operating system

Dual Mode Operation: kernel and user modes

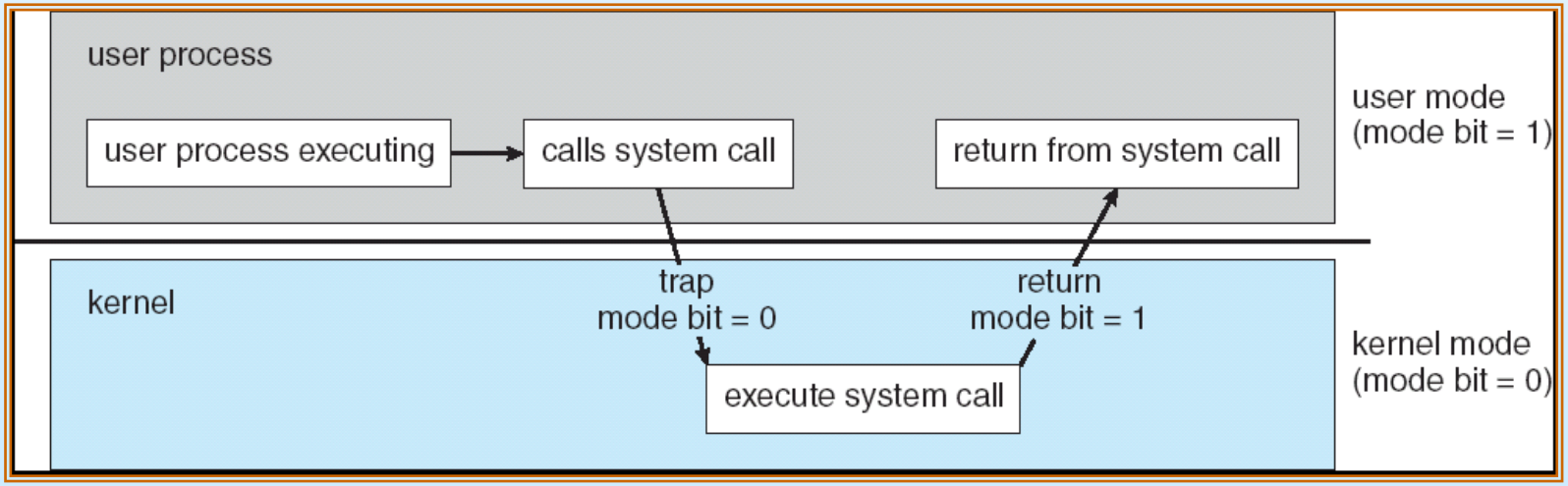
- **Dual-mode** operation allows OS to protect itself and other system components
- Every CPU hardware has a mode bit for:
  - **Kernel mode**
  - **User mode**
- **Mode bit** provided by **hardware**
  - Distinguish when system is running **user code** or **kernel code**
  - Some instructions designated as **privileged**
    - only executable by the kernel! (in kernel mode)
  - **System calls** change mode to kernel mode (by the OS)
  - Return from a system call restores user mode (by the OS)

- **Example:** what happens when a user executes the C function ??

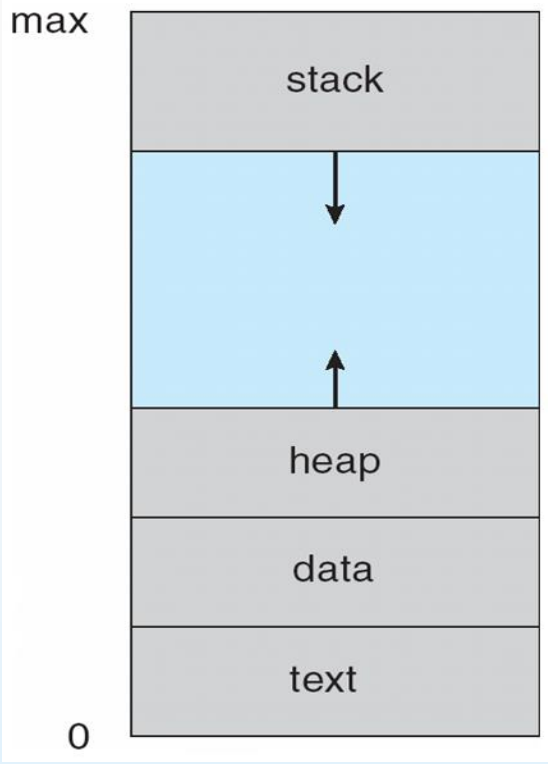
```
n = write(fd,buf,30)
```

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - When counter is zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

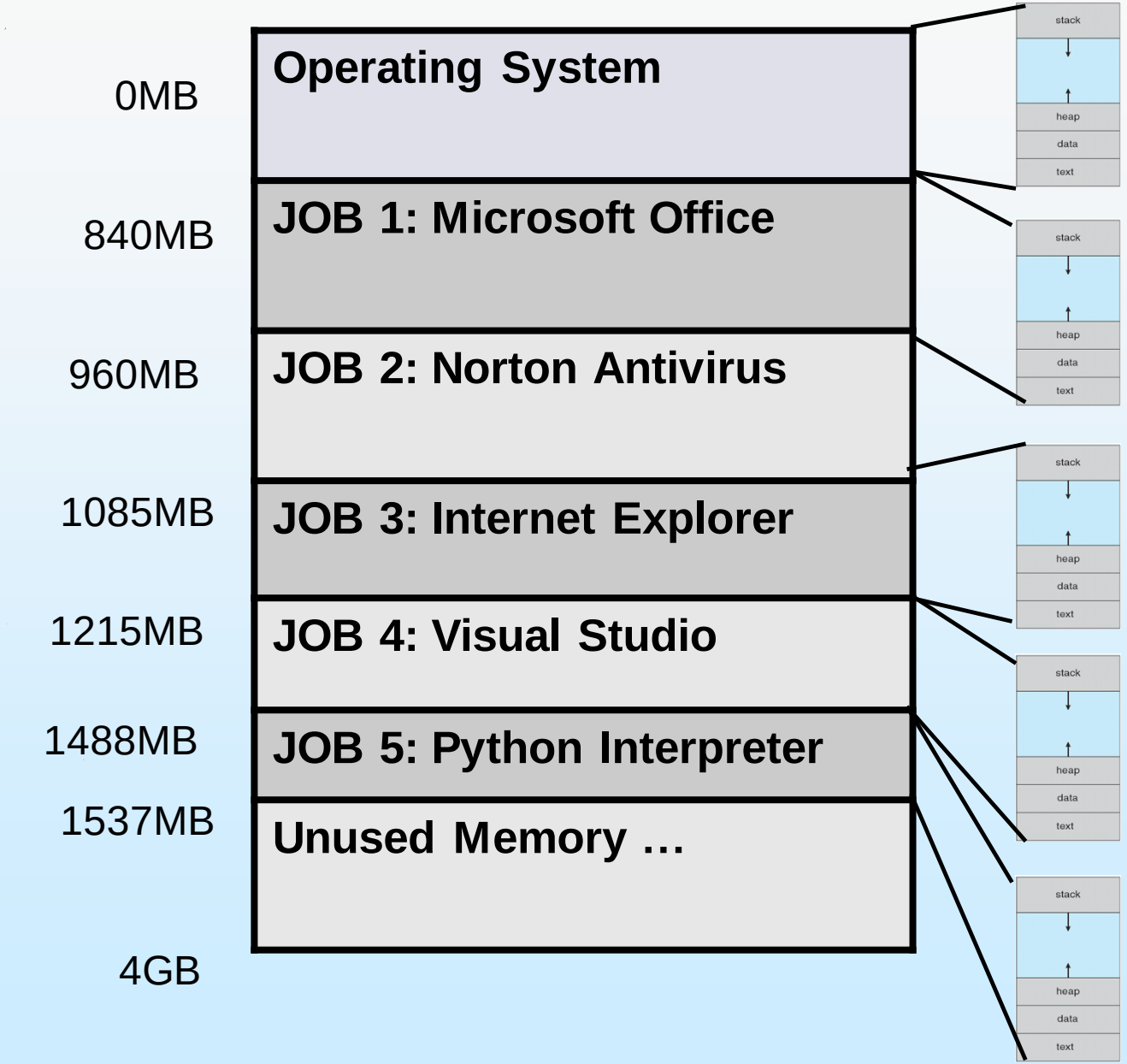


Process Management



- A process is **a program in execution**
- A process is a **unit of work** within the system
- A program is a ***passive entity***
- A process is an ***active entity***:
  - The **program** is usually stored in a disk
  - The **process** runs in **CPU** and **RAM** !
- Process needs resources to accomplish its task
  - **CPU**, **memory space**, **I/O**, **disk files, etc.**
- **Process termination** requires **reclaim** of any **reusable resources**
- Process executes instructions sequentially, one at a time, until completion:
  - The program is stored in the “text” part of the process
  - In addition, a process also has “data”, “heap”, and “stack”

Memory Layout for Multiprogrammed System (2)



Multi-threaded Process

- Single-threaded process has one **program counter** specifying location of next instruction to execute
- **Multi-threaded** process consists of **several Threads**
  - Example: **Microsoft Word** enables editing, dictionary check, and backup at the same time (possibly on different cores concurrently!)
- Each **Thread** has a separate
  - Program counter
  - Registers
  - Stack
- Typically the system has many processes, some user, some operating system running concurrently on one or more CPUs
- **True Concurrency** is achieved by running 4 or 8 processes on the 4 or 8 cores.
- This is however limited to only 4 or 8 processes/threads and the majority of processes still need to run sequentially

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**
  - Example: Video player reading frames from network
- Providing mechanisms for **process communication**
  - Example: Displaying PDF document (Acrobat Reader) in a Chrome Tab
- Providing mechanisms for **deadlock handling**
  - Chrome is waiting for Acrobat to exit, Acrobat is waiting for Chrome to release a file handle - DEADLOCK

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 de-allocating memory space as needed:
- Space requirements occur very frequently during process lifetime (using C **malloc** function repeatedly)
- Each request has to be given to the OS and the OS tries to find a contiguous memory space to fulfill this request until all memory is exhausted …

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**
  - 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
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed

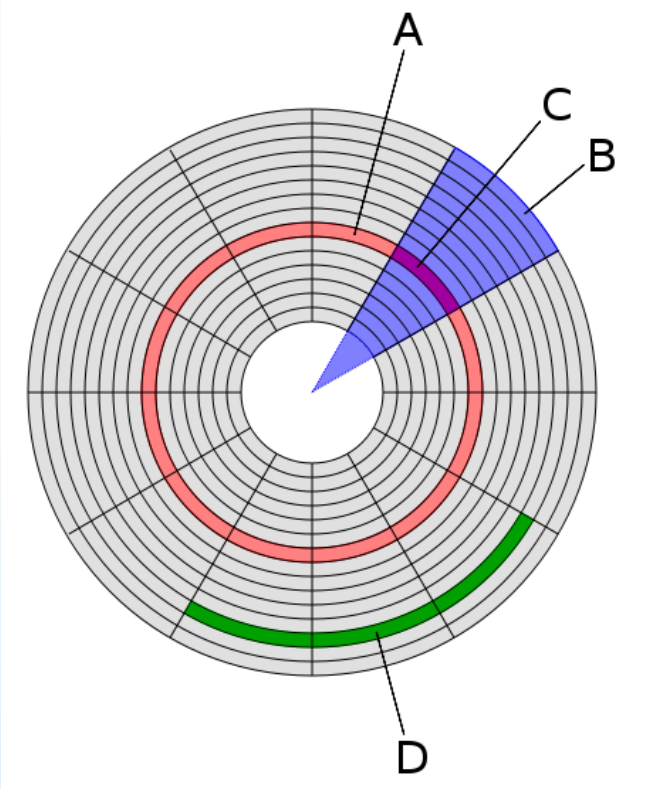
Hard Drive Structure

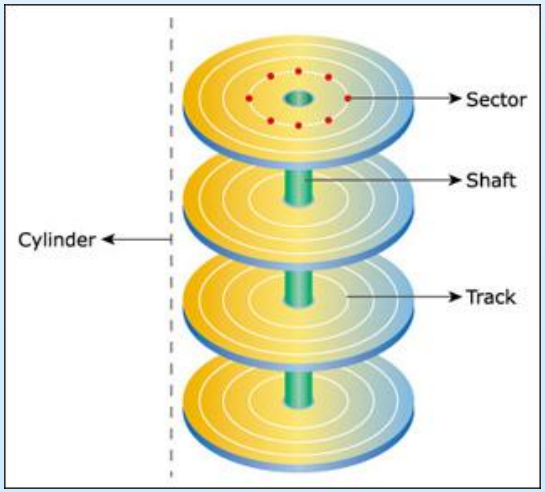
(A) Track
(B) *Geometrical* sector
(C) Track sector
(D) Cluster

Hard drive may consist of several disks (and heads) ordered in a cylinder

A file (like “oliver_twist.txt”) can be distributed across non-contiguous sectors, tracks, or even several disks!

Links:
http://en.wikipedia.org/wiki/Logical_block_addressing
http://www.tldp.org/LDP/tlk/dd/drivers.html





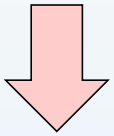
I/O Subsystem

- 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
  - Saves slow access to disk drives
- **Caching**
  storing parts of data in faster storage for performance
- **Spooling** (*simultaneous peripheral operations on-line)*
  the overlapping of output of one job with input of other jobs
- **General device-driver interface**
- **Drivers for specific hardware devices**

Read System Call

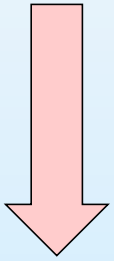
User Code:

```
fd = open("oliver_twist.txt", O_RDONLY, 0)
read(fd, buf, 65536)
```



Operating System Code:

```
Copy blocks 15,16,17, 32, 57, 321, ...
To memory buffer buf
[block = 4096 bytes, so need 16 blocks]
```



Device Driver:

Instruct the **DMA** to transfer the following sectors to memory (thru system bus)

Disk	**Cylinder**	**Sector**
5	14	C
3	6	A
8	3	H
2	12	D
…	…	…

When done interrupt the operating system
The operating system resumes control to user process

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 (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

Web-Based Computing

- Web has become ubiquitous
- PCs, tablets, smart phones interconnected
- More devices becoming networked to allow web access (iwatch, medical devices, wearable computing)
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows, client-side, have evolved into Linux and Windows XP/7/8, which can be clients and servers