Introduction

References:

   1. Abraham Silberschatz, Greg Gagne, and Peter Baer Galvin, "Operating System Concepts, **Ninth** Edition ", Chapter 1

Just as in ***The Blind Men and the Elephant***, this chapter looks at Operating Systems from a number of different viewpoints. No one view really shows the complete picture, but by looking from a number of different views, we can get a pretty good overall picture of what operating systems are all about.

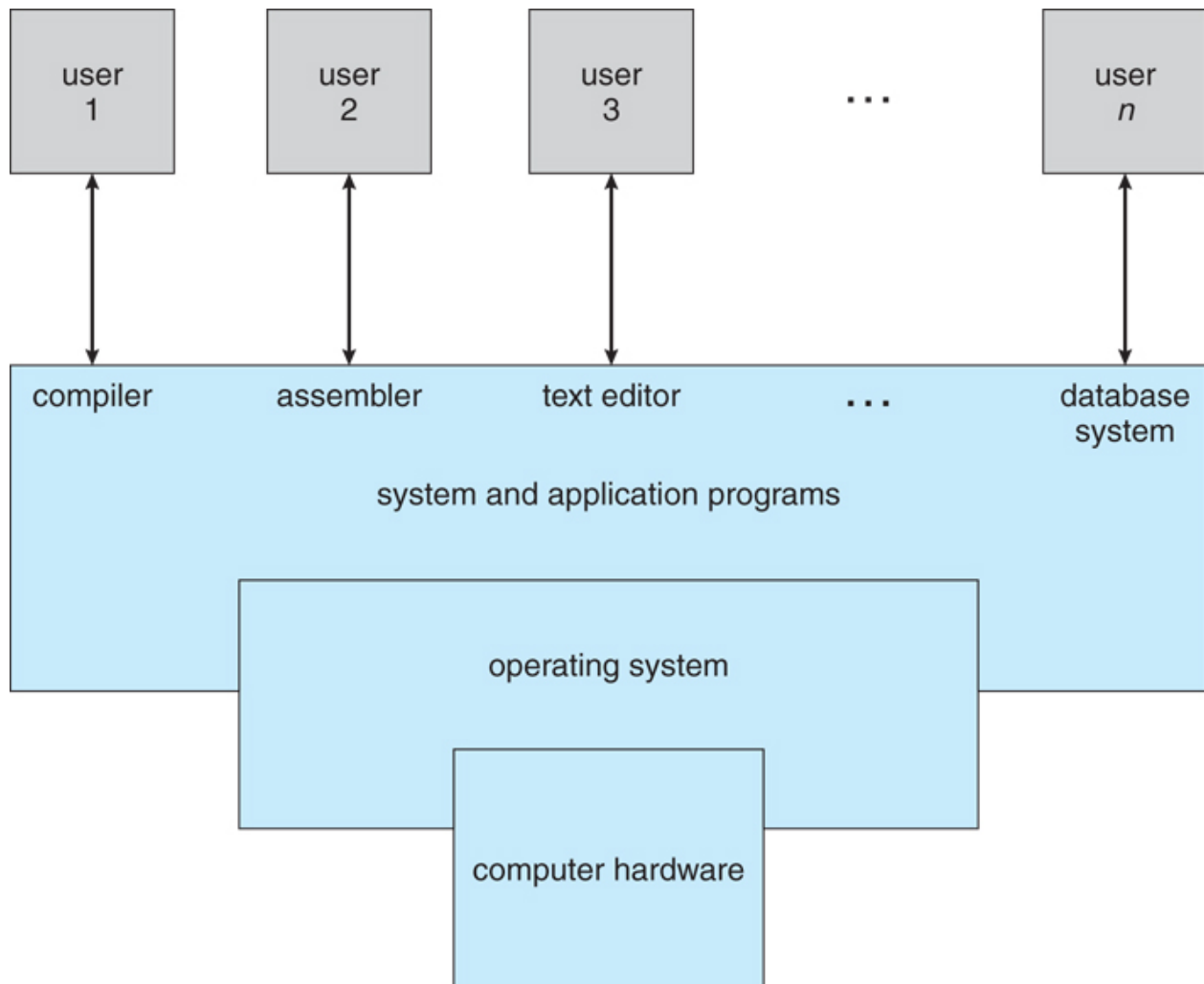## 1.1 What Operating Systems Do - For Users, For Applications, etc.

Figure 1.1 - Abstract view of the components of a computer system

- Computer = HW + OS + Apps + Users
- OS serves as interface between HW and (Apps & Users)
- OS provides services for Apps & Users
- OS manages resources (Government model, it doesn't produce anything.)
- Debates about what is included in the OS - Just the kernel, or everything the vendor ships?
  (Consider the distinction between system applications and 3rd party or user apps.)

1.2 Computer-System Organization - What are all the parts, and how do they fit together?
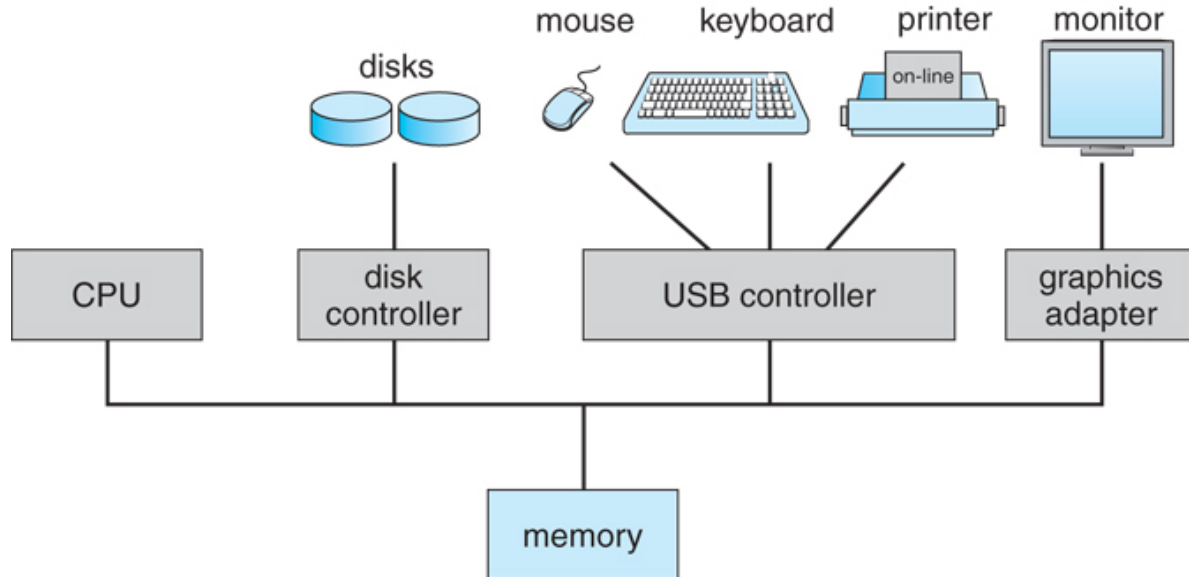
Figure 1.2 - A modern computer system

1.2.1 Computer-System Operation

- Bootstrap program
- Shared memory between CPU and I/O cards
- Time slicing for multi-process operation
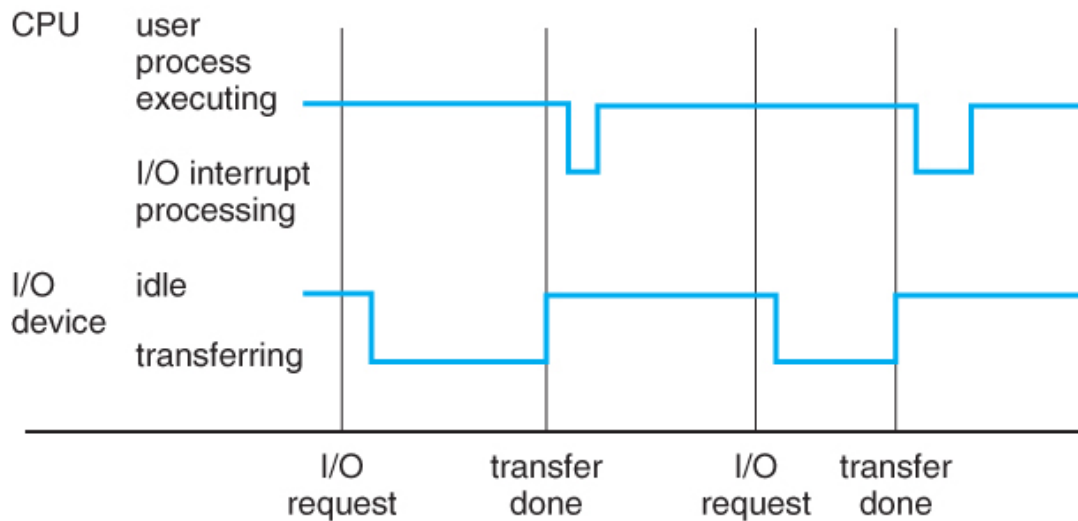- Interrupt handling - clock, HW, SW
- Implementation of system calls

Figure 1.3 - Interrupt timeline for a single process doing output

1.2.2 Storage Structure

- Main memory (RAM)
    - Programs must be loaded into RAM to run.
    - Instructions and data fetched from RAM into registers.
    - RAM is volatile
    - "Medium" size and speed
- Other electronic (volatile) memory is faster, smaller, and more expensive per bit:
    - Registers
    - CPU Cache
- Non-volatile memory ("permanent" storage) is slower, larger, and less expensive per bit:
    - Electronic disks
    - Magnetic disks
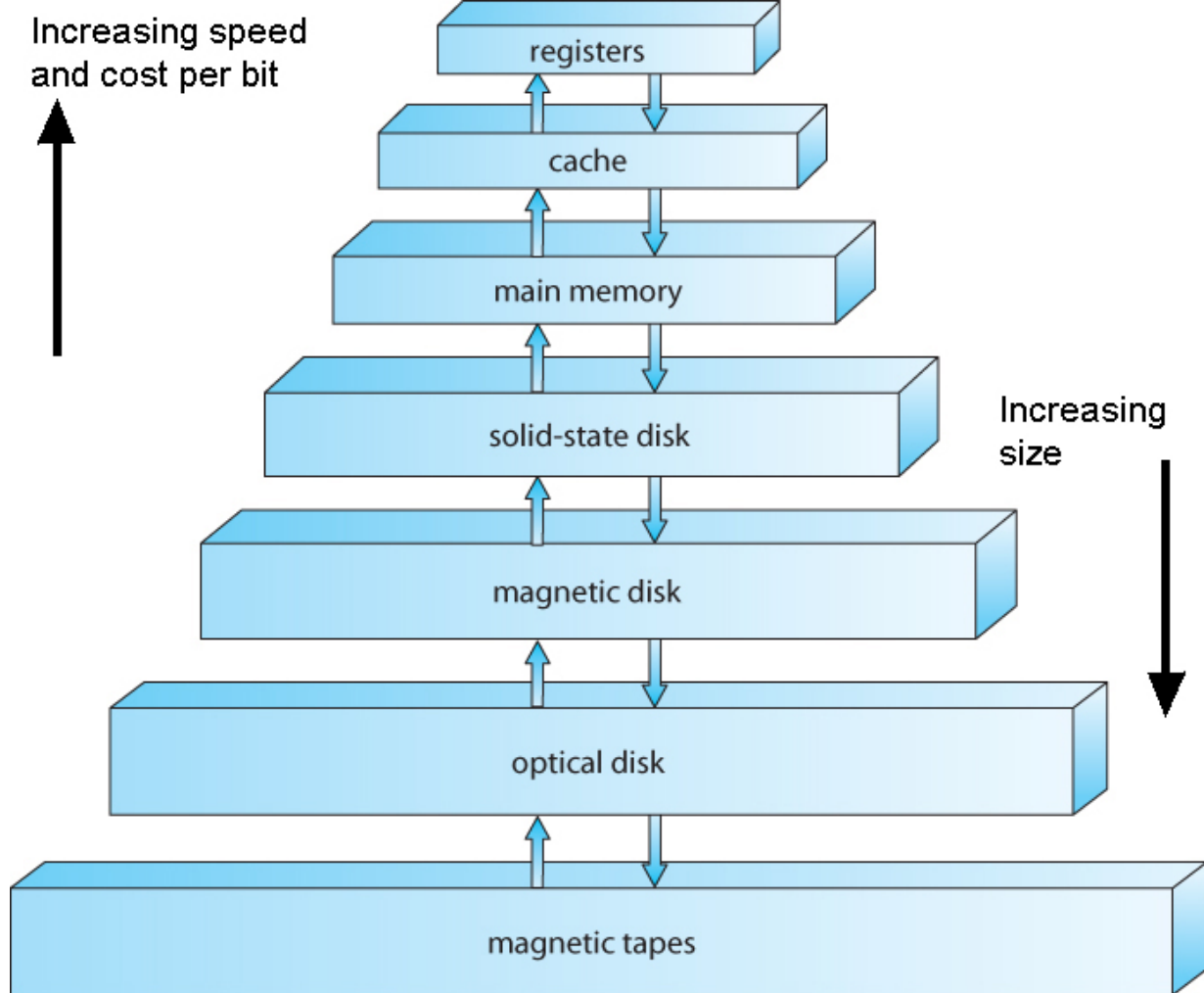   - Optical disks
    - Magnetic Tapes

Figure 1.4 - Storage-device hierarchy

1.2.3 I/O Structure

- Typical operation involves I/O requests, direct memory access (DMA), and interrupt handling.
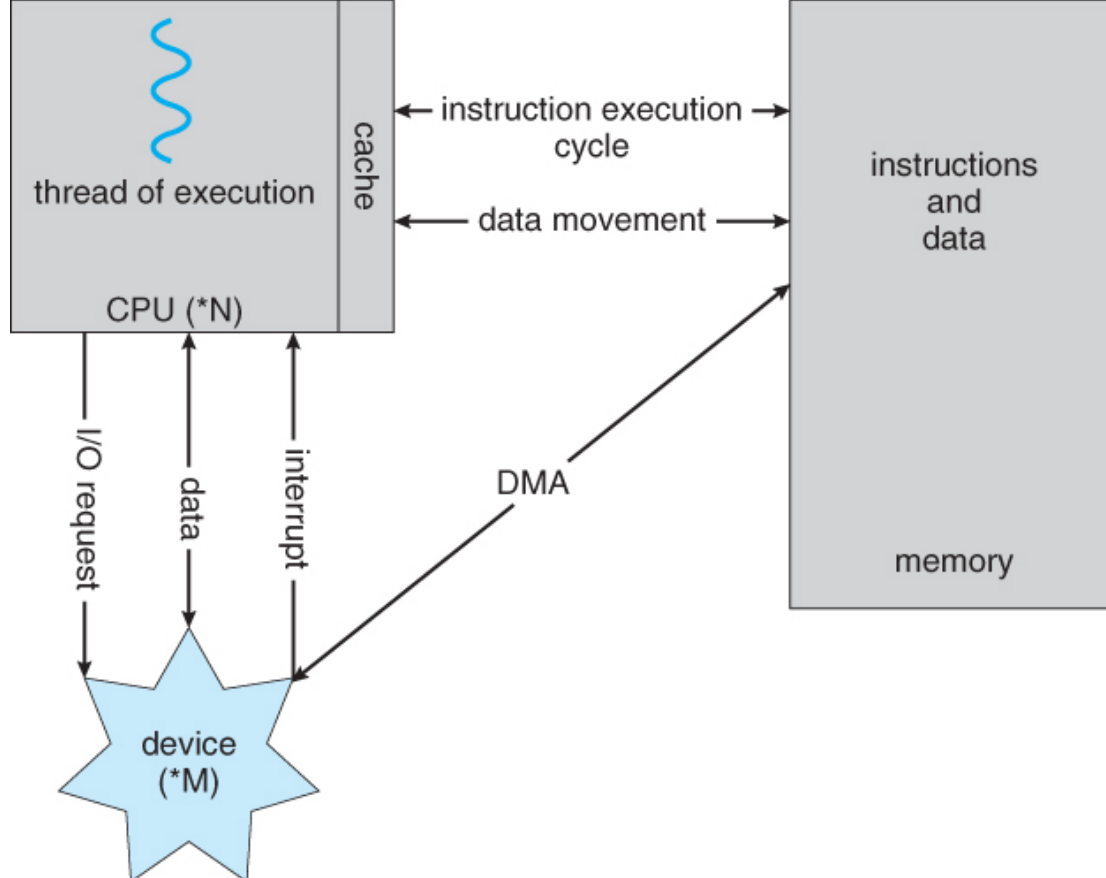
Figure 1.5 - How a modern computer system works

1.3 Computer-System Architecture - Different Operating Systems for Different Kinds of Computer Environments

1.3.1 Single-Processor Systems

- One main CPU which manages the computer and runs user apps.
- Other specialized processors (disk controllers, GPUs, etc.) do not run user apps.

1.3.2 Multiprocessor Systems

1. Increased throughput - Faster execution, but not 100% linear speedup.
2. Economy of scale - Peripherals, disks, memory, shared among processors.
3. Increased reliability
    - Failure of a CPU slows system, doesn't crash it.
    - Redundant processing provides system of checks and balances. (e.g. NASA)
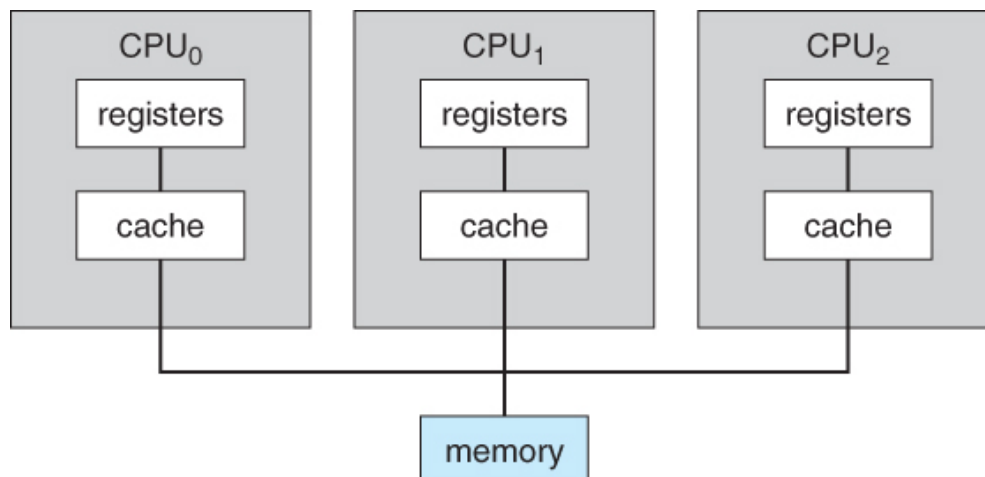

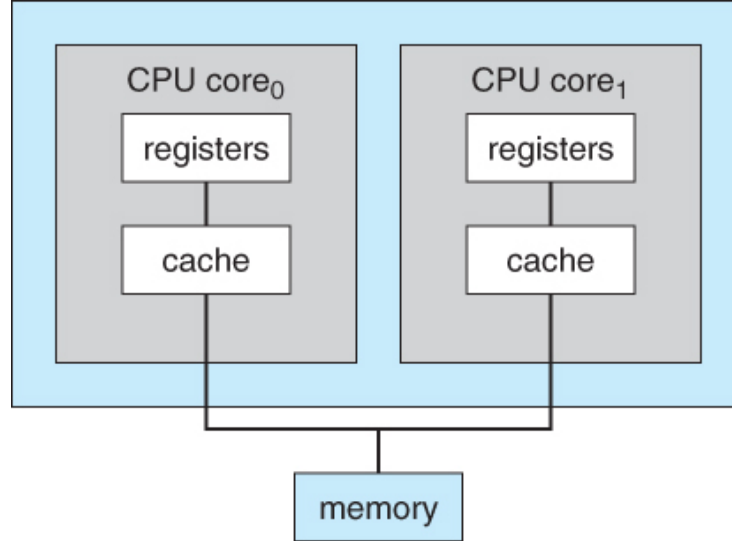**Figure 1.6 - Symmetric multiprocessing architecture**

Figure 1.7 - A dual-core design with two cores placed on the same chip

1.3.3 Clustered Systems

- Independent systems, with shared common storage and connected by a high-speed LAN, working together.
- Special considerations for access to shared storage are required, (Distributed lock management), as are collaboration protocols.
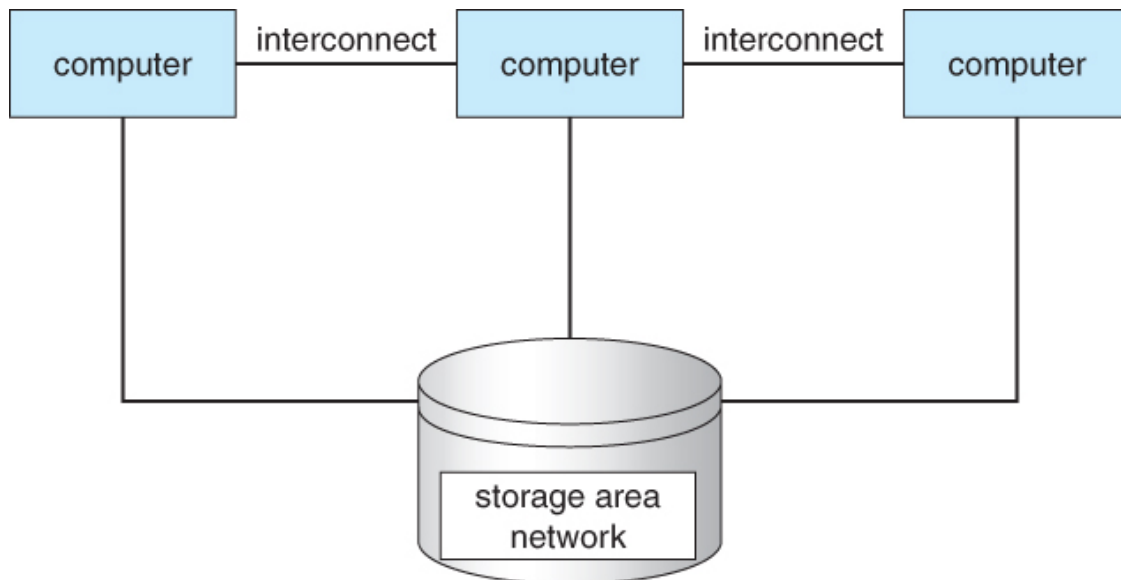

Figure 1.8 - General structure of a clustered system

1.4 Operating-System Structure

A time-sharing (multi-user multi-tasking) OS requires:

- Memory management
- Process management
- Job scheduling
- Resource allocation strategies
- Swap space / virtual memory in physical memory
- Interrupt handling
- File system management
- Protection and security
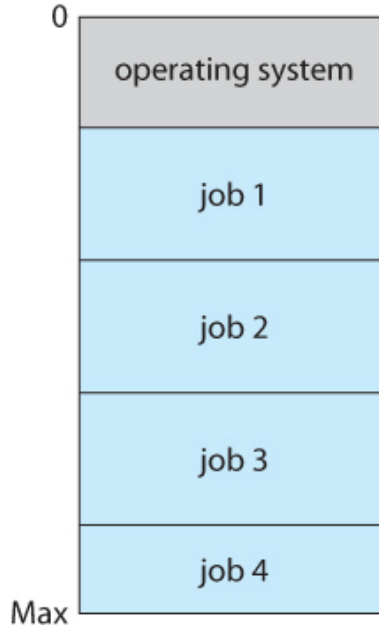- Inter-process communications

Figure 1.9 - Memory layout for a multiprogramming system

1.5 Operating-System Operations

Interrupt-driven nature of modern OSes requires that erroneous processes not be able to disturb anything else.

1.5.1 Dual-Mode and Multimode Operation

- User mode when executing harmless code in user applications
- Kernel mode (a.k.a. system mode, supervisor mode, privileged mode) when executing potentially dangerous code in the system kernel.
- Certain machine instructions (privileged instructions) can only be executed in kernel mode.
- Kernel mode can only be entered by making system calls. User code cannot flip the mode switch.
- Modern computers support dual-mode operation in hardware, and therefore most modern OSes support dual-mode operation.
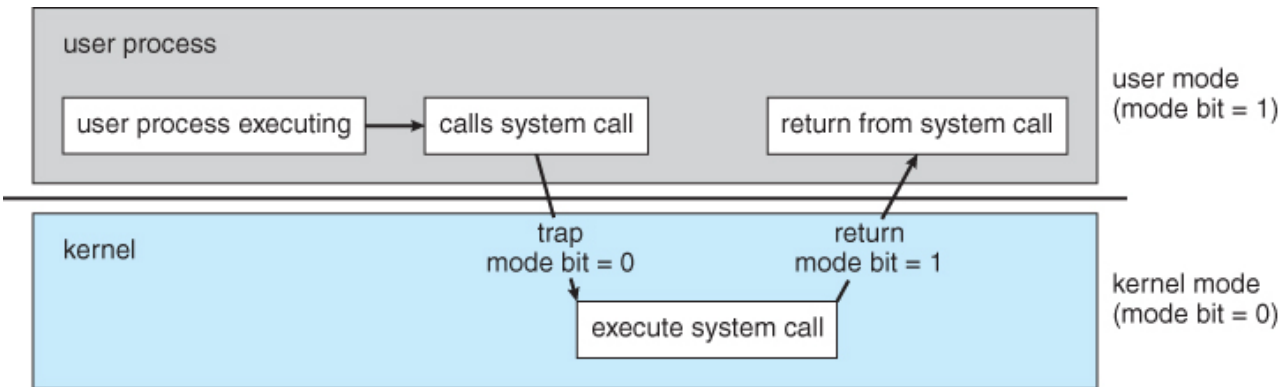

Figure 1.10 - Transition from user to kernel mode

- The concept of modes can be extended beyond two, requiring more than a single mode bit
- CPUs that support virtualization use one of these extra bits to indicate when the virtual machine manager, VMM, is in control of the system. The VMM has more privileges than ordinary user programs, but not so many as the full kernel.
- System calls are typically implemented in the form of software interrupts, which causes the hardware's interrupt handler to transfer control over to an appropriate interrupt handler, which is part of the operating system, switching the mode bit to kernel mode in the process. The interrupt handler checks exactly which interrupt was generated, checks additional parameters (generally passed through registers) if appropriate, and then calls the appropriate kernel service routine to handle the service requested by the system call.
- User programs' attempts to execute illegal instructions (privileged or non-existent instructions), or to access forbidden memory areas, also generate software interrupts, which are trapped by the interrupt handler and control is transferred to the OS, which issues an appropriate error message, possibly dumps data to a log (core) file for later analysis, and then terminates the offending program.

1.5.2 Timer

- Before the kernel begins executing user code, a timer is set to generate an interrupt.
- The timer interrupt handler reverts control back to the kernel.
- This assures that no user process can take over the system.
- Timer control is a privileged instruction, (requiring kernel mode.)

1.6 Process Management

An OS is responsible for the following tasks with regards to process management:

- Creating and deleting both user and system processes
- Ensuring that each process receives its necessary resources, without interfering with other processes.
- Suspending and resuming processes
- Process synchronization and communication
- Deadlock handling

1.7 Memory Management

An OS is responsible for the following tasks with regards to memory management:

- Keeping track of which blocks of memory are currently in use, and by which processes.
- Determining which blocks of code and data to move into and out of memory, and when.
- Allocating and deallocating memory as needed. (E.g. new, malloc)

1.8 Storage Management

1.8.1 File-System Management

An OS is responsible for the following tasks with regards to filesystem management:

- Creating and deleting files and directories
- Supporting primitives for manipulating files and directories. (open, flush, etc.)
- Mapping files onto secondary storage.
- Backing up files onto stable permanent storage media.

1.8.2 Mass-Storage Management

An OS is responsible for the following tasks with regards to mass-storage management:

- Free disk space management
- Storage allocation
- Disk scheduling

Note the trade-offs regarding size, speed, longevity, security, and re-writability between different mass storage devices, including floppy disks, hard disks, tape drives, CDs, DVDs, etc.

1.8.3 Caching

- There are many cases in which a smaller higher-speed storage space serves as a cache, or temporary storage, for some of the most frequently needed portions of larger slower storage areas.
- The hierarchy of memory storage ranges from CPU registers to hard drives and external storage. (See table below.)
- The OS is responsible for determining what information to store in what level of cache, and when to transfer data from one level to another.
- The proper choice of cache management can have a profound impact on system performance.
- Data read in from disk follows a migration path from the hard drive to main memory, then to the CPU cache, and finally to the registers before it can be used, while data being written follows the reverse path. Each step (other than the registers) will typically fetch more data than is immediately needed, and cache the excess in order to satisfy future requests faster. For writing, small amounts of data are frequently buffered until there is enough to fill an entire "block" on the next output device in the chain.
- The issues get more complicated when multiple processes (or worse multiple computers) access common data, as it is important to ensure that every access reaches the most up-to-date copy of the cached data (amongst several copies in different cache levels.)

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

Figure 1.11 - Performance of various levels of storage
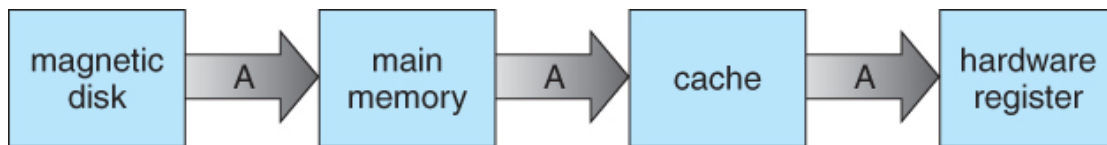

Figure 1.12 - Migration of integer A from disk to register

1.8.4 I/O Systems

The I/O subsystem consists of several components:

- A memory-management component that includes buffering, caching, and spooling.
- A general device-driver interface.
- Drivers for specific hardware devices.
- (UNIX implements multiple device interfaces for many types of devices, one for accessing the device character by character and one for accessing the device block by block. These can be seen by doing a long listing of /dev, and looking for a "c" or "b" in the first position. You will also note that the "size" field contains two numbers, known as the major and minor device numbers, instead of the normal one. The major number signifies which device driver handles I/O for this device, and the minor number is a parameter passed to the driver to let it know which specific device is being accessed. Where a device can be accessed as either a block or character device, the minor numbers for the two options usually differ by a single bit.)

1.9 Protection and Security

- **Protection** involves ensuring that no process access or interfere with resources to which they are not entitled, either by design or by accident. (E.g. "protection faults" when pointer variables are misused.)
- **Security** involves protecting the system from deliberate attacks, either from legitimate users of the system attempting to gain unauthorized access and privileges, or external attackers attempting to access or damage the system.

1.10 Kernel Data Structures

1.10.1 Lists, Stacks, and Queues
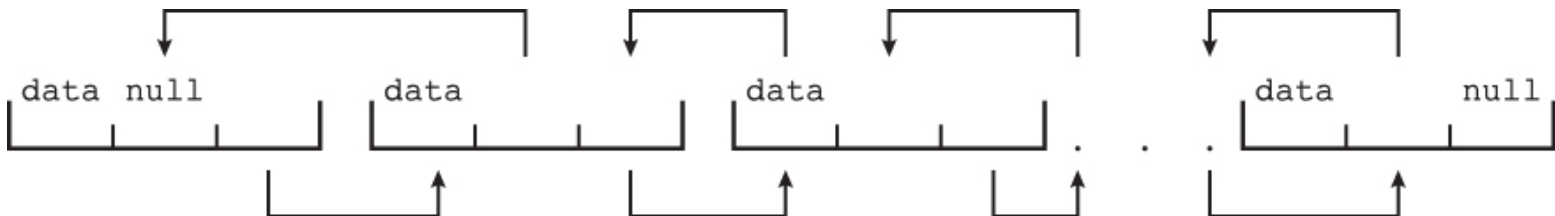

Figure 1.13 - Singly linked list

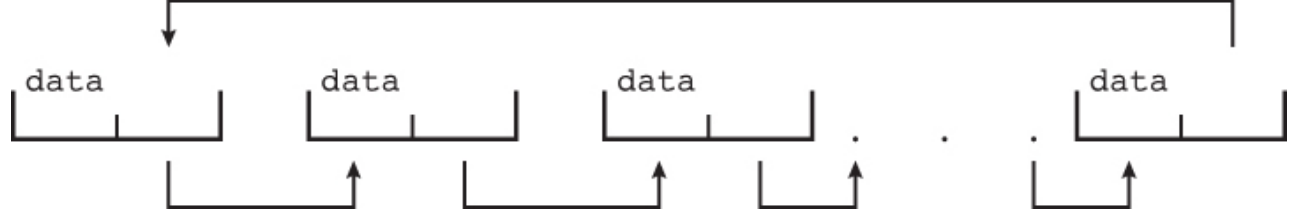
Figure 1.14 - Doubly linked list

Figure 1.15 - Circularly linked list
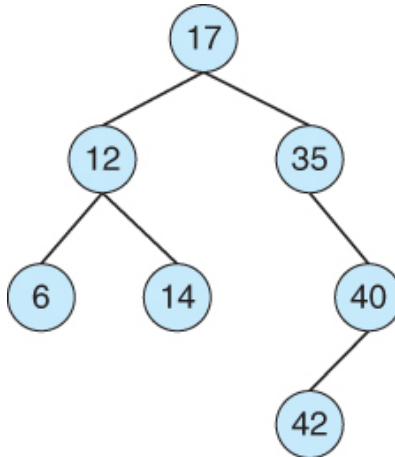
1.10.2 Trees

Figure 1.16 - Binary search trees
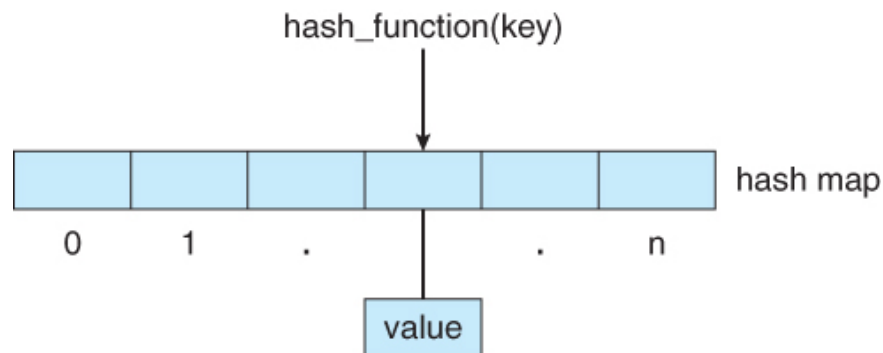
1.10.3 Hash Functions and Maps

Figure 1.17 - Hash map

1.10.4 Bitmaps

- A string of 1s and 0s used to keep track of the boolean state of a collection of objects, such as the free state of blocks on a disk or pages in memory..

1.11 Computing Environments

1.11.1 Traditional Computing

1.11.2 Mobile Computing

- Computing on small handheld devices such as smart phones or tablets. (As opposed to laptops, which still fall under traditional computing.)
- May take advantage of additional built-in sensors, such as GPS, tilt, compass, and inertial movement.
- Typically connect to the Internet using wireless networking (IEEE 802.11) or cellular telephone technology.
- Limited in storage capacity, memory capacity, and computing power relative to a PC.
- Generally uses slower processors, that consume less battery power and produce less heat.
- The two dominant OSes today are Google Android and Apple iOS.

1.11.3 Distributed Systems

- Distributed Systems consist of multiple, possibly heterogeneous, computers connected together via a network and cooperating in some way, form, or fashion.
- Networks may range from small tight LANs to broad reaching WANs.
  - WAN = Wide Area Network, such as an international corporation
  - MAN =Metropolitan Area Network, covering a region the size of a city for example.
  - LAN =Local Area Network, typical of a home, business, single-site corporation, or university campus.
  - PAN = Personal Area Network, such as the bluetooth connection between your PC, phone, headset, car, etc.
- Network access speeds, throughputs, reliabilities, are all important issues.
- OS view of the network may range from just a special form of file access to complex well-coordinated network operating systems.
- Shared resources may include files, CPU cycles, RAM, printers, and other resources.

1.11.4 Client-Server Computing

- A defined server provides services (HW or SW) to other systems which serve as clients. (Technically clients and servers are processes, not HW, and may co-exist on the same physical computer.)
- A process may act as both client and server of either the same or different resources.
- Served resources may include disk space, CPU cycles, time of day, IP name information, graphical displays (X Servers), or other resources.
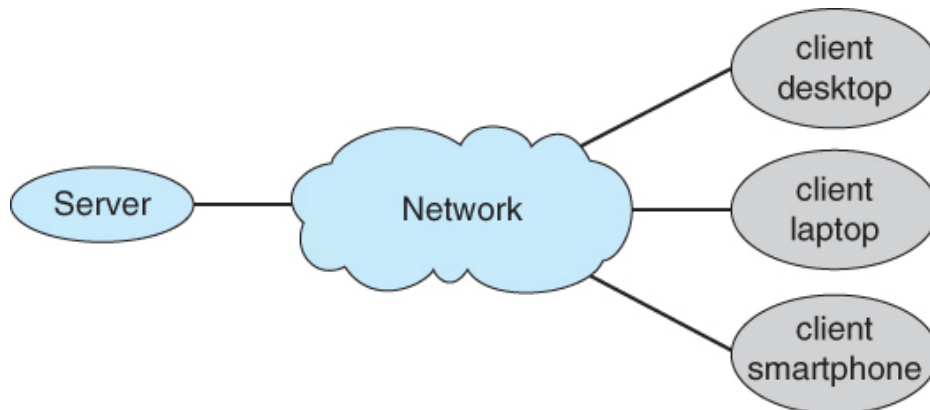

Figure 1.18 - General structure of a client-server system

1.11.5 Peer-to-Peer Computing

- Any computer or process on the network may provide services to any other which requests it. There is no clear "leader" or overall organization.
- May employ a central "directory" server for looking up the location of resources, or may use peer-to-peer searching to find resources.
- E.g. Skype uses a central server to locate a desired peer, and then further communication is peer to peer.
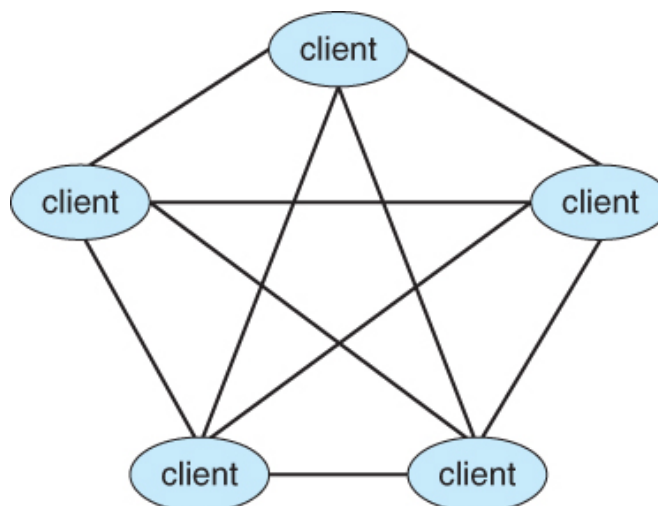

Figure 1.19 - Peer-to-peer system with no centralized service

1.11.6 Virtualization

- Allows one or more "guest" operating systems to run on virtual machines hosted by a single physical machine and the virtual machine manager.
- Useful for cross-platform development and support.

- For example, a student could run UNIX on a virtual machine, hosted by a virtual machine manager on a Windows based personal computer. The student would have full root access to the virtual machine, and if it crashed, the underlying Windows machine should be unaffected.
- System calls have to be caught by the VMM and translated into (different) system calls made to the real underlying OS.
- Virtualization can slow down program that have to run through the VMM, but can also speed up some things if virtual hardware can be accessed through a cache instead of a physical device.
- Depending on the implementation, programs can also run simultaneously on the native OS, bypassing the virtual machines.
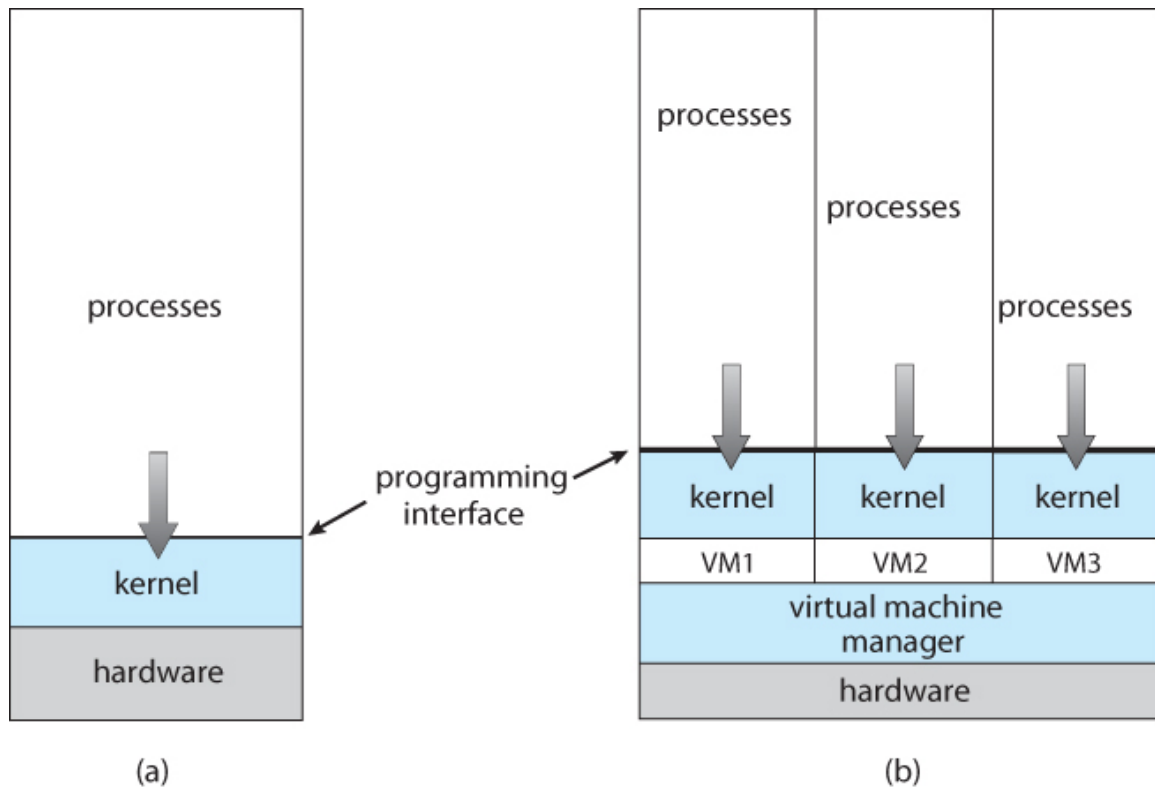

Figure 1.20 - VMWare

- To run Linux on a Windows system using VMware, follow these steps:
    1. Download the free "VMware Player" tool from http://www.vmware.com/download/player and install it on your system
    2. Choose a Linux version from among hundreds of virtual machine images at http://www.vmware.com/appliances
    3. Boot the virtual machine within VMware Player.

1.11.7 Cloud Computing

- Delivers computing, storage, and applications as a service over a network.
- Types of cloud computing:
    - Public cloud - Available to anyone willing to pay for the service.
    - Private cloud - Run by a company for internal use only.
    - Hybrid cloud - A cloud with both public and private components.
    - Software as a Service - SaaS - Applications such as word processors available via the Internet
    - Platform as a Service - PaaS - A software stack available for application use, such as a database server
    - Infrastructure as a Service - IaaS - Servers or storage available on the Internet, such as backup servers, photo storage, or file storage.
    - Service providers may provide more than one type of service
- Clouds may contain thousands of physical computers, millions of virtual ones, and petabytes of total storage.
- Web hosting services may offer (one or more) virtual machine(s) to each of their clients.
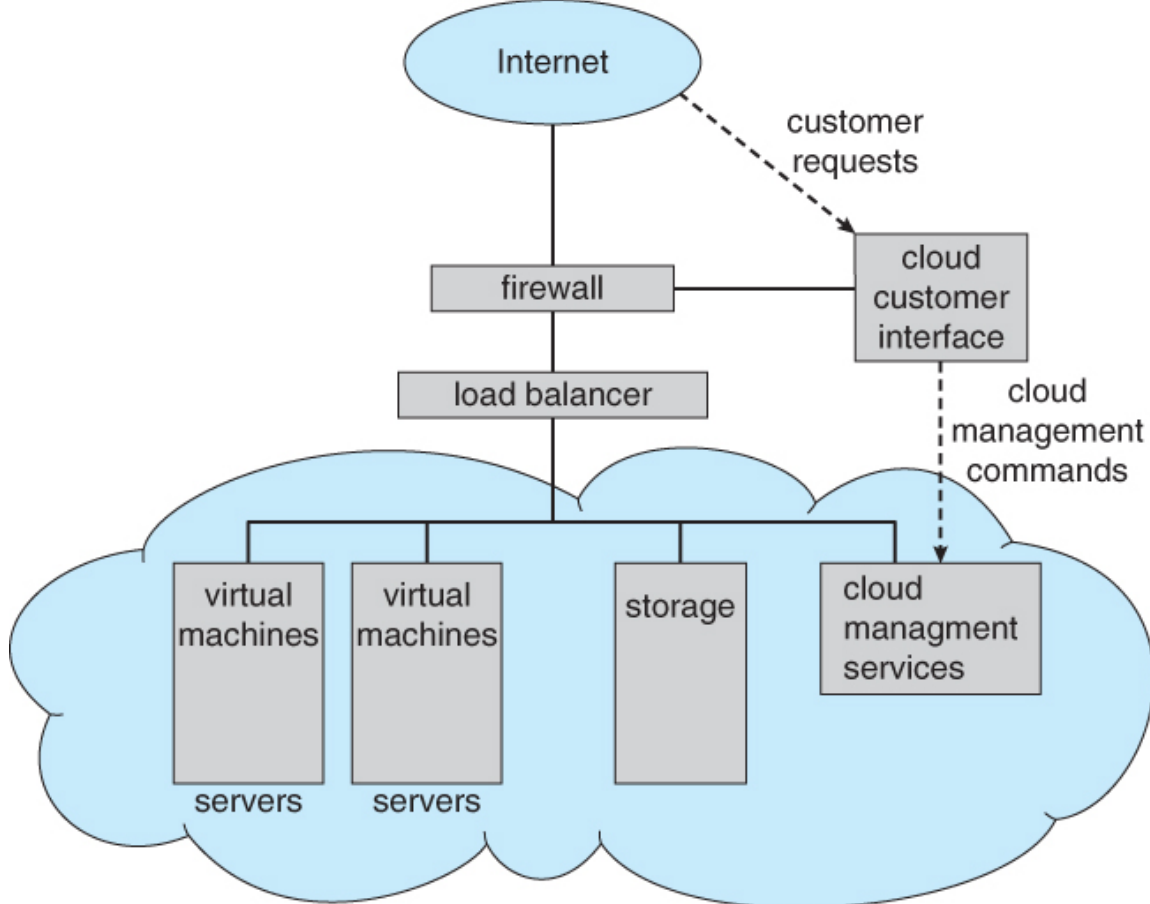
Figure 1.21 - Cloud computing

1.11.8 Real-Time Embedded Systems

- Embedded into devices such as automobiles, climate control systems, process control, and even toasters and refrigerators.
- May involve specialized chips, or generic CPUs applied to a particular task. (Consider the current price of 80286 or even 8086 or 8088 chips, which are still plenty powerful enough for simple electronic devices such as kids toys.)
- Process control devices require real-time (interrupt driven) OSes. Response time can be critical for many such devices.

1.12 Open-Source Operating Systems

- (For more information on the Flourish conference held at UIC on the subject of Free Libre and Open Source Software , visit http://www.flourishconf.com)
- Open-Source software is published (sometimes sold) with the source code, so that anyone can see and optionally modify the code.
- Open-source SW is often developed and maintained by a small army of loosely connected often unpaid programmers, each working towards the common good.
- Critics argue that open-source SW can be buggy, but proponents counter that bugs are found and fixed quickly, since there are so many pairs of eyes inspecting all the code.
- Open-source operating systems are a good resource for studying OS development, since students can examine the source code and even change it and re-compile the changes.

1.12.1 History

- At one time (1950s) a lot of code was open-source.
- Later, companies tried to protect the privacy of their code, particularly sensitive issues such as copyright protection algorithms.
- In 1983 Richard Stallman started the GNU project to produce an open-source UNIX.
- He later published the GNU Manifesto, arguing that ALL software should be open-source, and founded the Free Software Foundation to promote open-source development.
- FSF and GNU use the GNU General Public License which essentially states that all users of the software have full rights to copy and change the SW however they wish, so long as anything they distribute further contain the same license agreement. (Copylefting)

1.12.2 Linux

- Developed by Linus Torvalds in Finland in 1991 as the first full operating system developed by GNU.

- Many different distributions of Linux have evolved from Linus's original, including RedHat, SUSE, Fedora, Debian, Slackware, and Ubuntu, each geared toward a different group of end-users and operating environments.
- To run Linux on a Windows system using VMware, follow these steps:
    1. Download the free "VMware Player" tool from http://www.vmware.com/download/player and install it on your system
    2. Choose a Linux version from among hundreds of virtual machine images at http://www.vmware.com/appliances
    3. Boot the virtual machine within VMware Player.

1.12.3 BSD UNIX

- UNIX was originally developed at ATT Bell labs, and the source code made available to computer science students at many universities, including the University of California at Berkeley, UCB.
- UCB students developed UNIX further, and released their product as BSD UNIX in both binary and source-code format.
- BSD UNIX is not open-source, however, because a license is still needed from ATT.
- In spite of various lawsuits, there are now several versions of BSD UNIX, including Free BSD, NetBSD, OpenBSD, and DragonflyBSD
- The source code is located in /usr/src.
- The core of the Mac operating system is Darwin, derived from BSD UNIX, and is available at http://developer.apple.com/opensource/index.html

1.13.4 Solaris

- Solaris is the UNIX operating system for computers from Sun Microsystems.
- Solaris was originally based on BSD UNIX, and has since migrated to ATT SystemV as its basis.
- Parts of Solaris are now open-source, and some are not because they are still covered by ATT copyrights.
- It is possible to change the open-source components of Solaris, re-compile them, and then link them in with binary libraries of the copyrighted portions of Solaris.
- Open Solaris is available from http://www.opensolaris.org/os/
- Solaris also allows viewing of the source code online, without having to download and unpack the entire package.

1.13.5 Utility

- The free software movement is gaining rapidly in popularity, leading to thousands of ongoing projects involving untold numbers of programmers.
- Sites such as http://freshmeat.net/ and http://distrowatch.com/ provide portals to many of these projects.

1.14 Summary