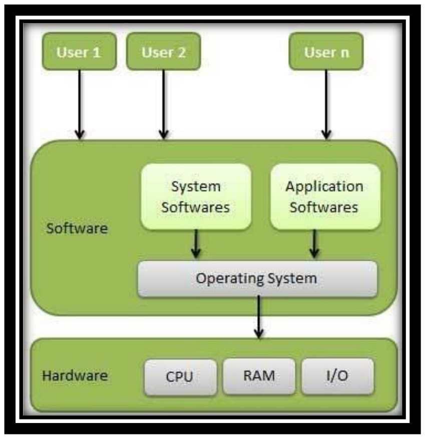
Unit 1
Chapter 1: Introduction

<u>Introduction:</u>

An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

<u>Definition:</u>

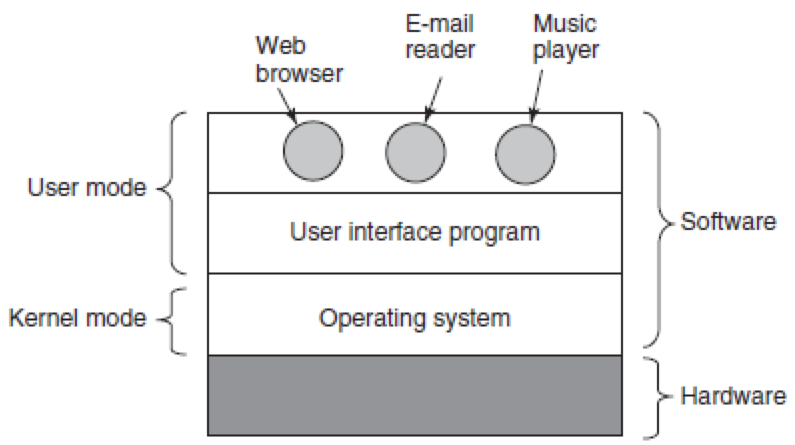
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

A simple overview of the main components under discussion here is given in Fig. Here we see the hardware at the bottom. The hardware consists of chips, boards, disks, a keyboard, a monitor, and similar physical objects. On top of the hardware is the software. Most computers have two modes of operation: kernel mode and user mode. The operating system, the most fundamental piece of software, runs in kernel mode (also called supervisor mode). In this mode it has complete access to all the hardware and can execute any instruction the machine is capable of executing. The rest of the software runs in user mode, in which only a subset of the machine instructions is available. In particular, those instructions that affect control of the machine or do I/O)Input/Output" are forbidden to user-mode programs. We will come back to the difference between kernel mode and user mode repeatedly throughout this book. It plays a crucial role in how operating systems work.



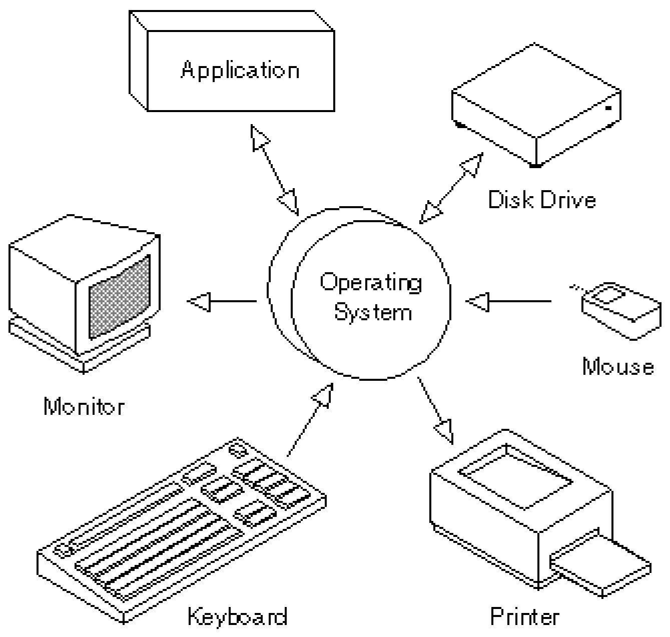
The user interface program, shell or GUI, is the lowest level of user-mode software, and allows the user to start other programs, such as a Web browser, email reader, or music player. These programs, too, make heavy use of the operating system.

<u>WHAT IS AN OPERATING SYSTEM?</u>

An operating system or OS is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer operating system, a computer and software programs would be useless. The picture is an example of Microsoft Windows XP, a popular operating system and what the box may look like if you were to visit a local retail store to purchase it. When computers were first introduced, the user interacted with them using a command line interface, which required commands. Today, almost every computer is using a Graphical User Interface (GUI) operating system that is much easier to use and operate.
Examples of computer operating systems Microsoft Windows 7 - PC and IBM compatible operating system. Windows is the most common and used operating system. Apple MacOS - Apple Mac operating system. The only Apple computer operating system

is MacOS. Ubuntu Linux - A popular variant of Linux used with PC and IBM compatible computers. Google Android - operating system used with Android compatible phones. iOS - Operating system used with the Apple iPhone.

Computer operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as printers.
For large systems, the operating system has even greater responsibilities and powers. It is like a traffic cop -- it makes sure those different programs and users running at the same time do not interfere with each other. The operating system is also responsible for *security*, ensuring that unauthorized users do not access the system.



Classification of Operating systems

- **Multi-user:** Allows two or more users to run programs at the same time. Some operating systems permit hundreds or even thousands of concurrent users.
- **Multiprocessing :** Supports running a program on more than one CPU.
- **Multitasking :** Allows more than one program to run concurrently.
- **Multithreading :** Allows different parts of a single program to run concurrently.
- **Real time:** Responds to input instantly. General-purpose operating systems, such as DOS and UNIX, are not real-time.

Operating systems provide a software platform on top of which other programs, called *application programs,* can run. The application programs must be written to run on top of a particular operating system. Your choice of operating system, therefore, determines to a great extent the applications you can run. For PCs, the most popular operating systems are DOS, OS/2, and Windows, but others are available, such as Linux.

Interacting With the Operating System

As a user, you normally interact with the operating system through a set of commands. For example, the DOS operating system contains commands such as COPY and RENAME for copying files and changing the names of files, respectively. The commands are accepted and executed by a part of the operating system called the command processor or command line interpreter. Graphical user interfaces allow you to enter commands by pointing and clicking at objects that appear on the screen.

History of Operating System

Historically operating systems have been tightly related to the computer architecture, it is good idea to study the history of operating systems from the architecture of the computers on which they run. Operating systems have evolved through a number of distinct phases or generations which corresponds roughly to the decades.

The 1940's - First Generations

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages). Operating systems were unheard of.

The 1950's - Second Generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which

several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.
For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

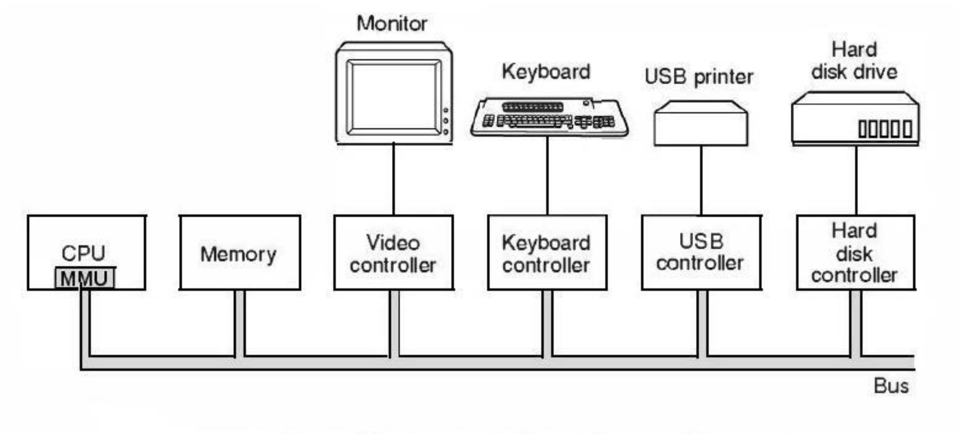
Another major feature in third-generation operating system was the technique called spooling (simultaneous peripheral operations on line). In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output. Instead of writing directly to a printer, for example, outputs are written to the disk. Programs can run to completion faster, and other programs can be initiated sooner when the printer becomes available, the outputs may be printed.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it become possible to build desktop computers as powerful as the mainframes of the 1970s. Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC and other machines using the Intel 8088 CPU and its successors, and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

COMPUTER HARDWARE REVIEW

An operating system is closely attached to the hardware of the computer it runs on. It expands the computer's instruction set and controls its resources. To work, it must know a great deal about the hardware, at least about how the hardware appears to the programmer. Therefore, let us review computer hardware in brief as found in modern personal computers. After that, we can start getting into the details of what operating systems do and how they work. Theoretically, a simple personal computer can be abstracted to a model resembling that of following figure. The CPU, memory, and I/O devices are all joined by a system bus and communicate with one another over it. Modern personal computers have a more complex structure, involving several buses, which we will look at later. For the time being, this model will be adequate. In the following sections, we will briefly review these components and look at some of the hardware problems that are of concern to operating system designers. Unnecessary to say, this will be a very compact summary.



Some of the components of a simple personal computer.

Processors

- The "brain" of the computer is the CPU. It brings instructions from memory and carries out them. The fundamental cycle of every CPU is to bring the first instruction from memory, interpret it to decide its type and operands, execute it, and then bring, decode, and carry out following instructions. The cycle is repeated until the program comes to an end. Thus, programs are executed.
- Each CPU has a particular set of instructions that it can implement. Because accessing memory to get an instruction or data word takes much longer than executing an instruction, all CPUs include some registers inside to hold key variables and temporary results. Therefore the instruction set generally contains instructions to load a word from memory into a register, and store a word from a register into memory. Other instructions merge two operands from registers, memory, or both into a result, such as adding two words and storing the result in a register or in memory.
- In addition to the general registers used to hold variables and temporary results, most computers have some special registers that are visible to the programmer. One of these is the program counter, which contains the memory address of the next instruction to be obtained. After that instruction has been obtained, the program counter is updated to point to its successor.
- Another register is the stack pointer, which points to the top of the current stack in memory. The stack includes one frame for each procedure that has been entered but not yet exited. A procedure's stack frame holds those input parameters, local variables, and temporary variables that are not kept in registers.
- Yet another register is the PSW (Program Status Word). This register includes the condition code bits, which are set by comparison instructions, the CPU priority, the mode (user or kernel), and various other control bits. User programs may

normally read the entire PSW but normally may write only some of its fields. The PSW plays an important role in system calls and I/0.

- The operating system must be well-informed of all the registers. When time multiplexing the CPU, the operating system will frequently stop the running program to (re)start another one. Every time it stops a running program, the operating system must save all the registers so they can be restored when the program runs later.
- Even more advanced than a pipeline design is a superscalar CPU, shown in the following figure 1(b). In this design, multiple execution units are present, for instance, one for integer arithmetic, one for floating-point arithmetic, and one for Boolean operations. Two or more instructions are fetched at once, decoded, and dumped into a

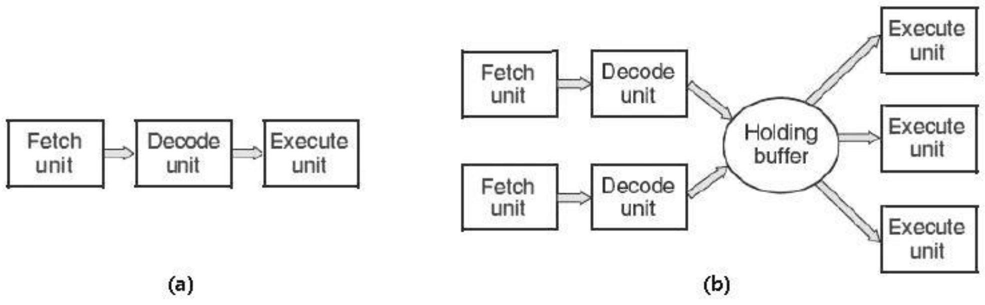


Figure 1. (a) A three-stage pipeline. (b) A superscalar CPU.

holding buffer until they can be implemented. As soon as an execution unit is free, it looks in the holding buffer to see if there is an instruction it can handle, and if so, it removes the instruction from the buffer and carries out it. An implication of this design is that program instructions are often executed out of order. Primarily, it is up to the hardware to make sure the result produced is the same one a sequential implementation would have produced, but an annoying amount of the complication is imposed onto the operating system, as we shall see.

Multithreaded and Multicore Chips

- According to Moore's law, the number of transistors on a chip doubles every 18 months. This "law" is not like any law of physics, like conservation of momentum, but is an observation by Intel cofounder Gordon Moore of how fast process engineers at the semiconductor companies are able to shrink their transistors. Moore's law has held for three decades now and is expected to hold for at least one more.
- The large quantity of transistors is leading to a problem: what to do with all of them? We saw one approach above: superscalar architectures, with several functional units. But as the number of transistors increases, even more is

possible. One clear thing to do is put bigger caches on the CPU chip and that is definitely happening, but ultimately the point of diminishing returns is reached.

- Multithreading has implications for the operating system because each thread appears to the operating system as a separate CPU. Think of a system with two actual CPUs, each with two threads. The operating system will see this as four CPUs. If there is only enough work to keep two CPUs busy at a certain point in time, it may unintentionally schedule two threads on the same CPU, with the other CPU completely idle. This choice is far less efficient than using one thread on each CPU. The successor to the Pentium 4, the Core (also Core 2) architecture does not have hyper threading, but Intel has announced that the Core's successor will have it again.
- Further than multithreading, we have CPU chips with two or four or more complete processors or cores on them. The multicore chips of the following figure effectively carry four minichips on them, each with its own independent CPU. (The caches will be explained below.) Making use of such a multicore chip will definitely need a multiprocessor operating system.

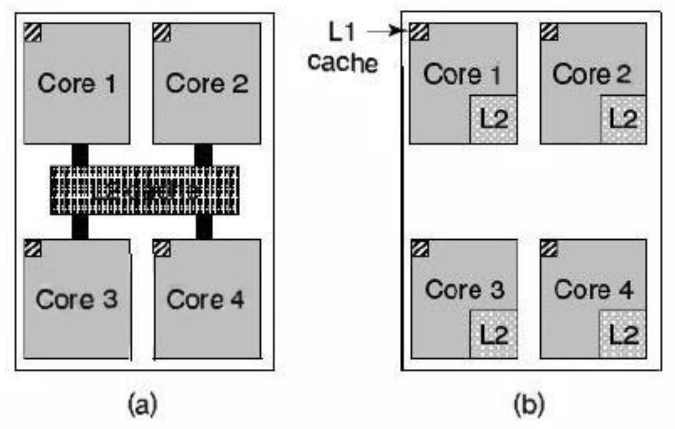
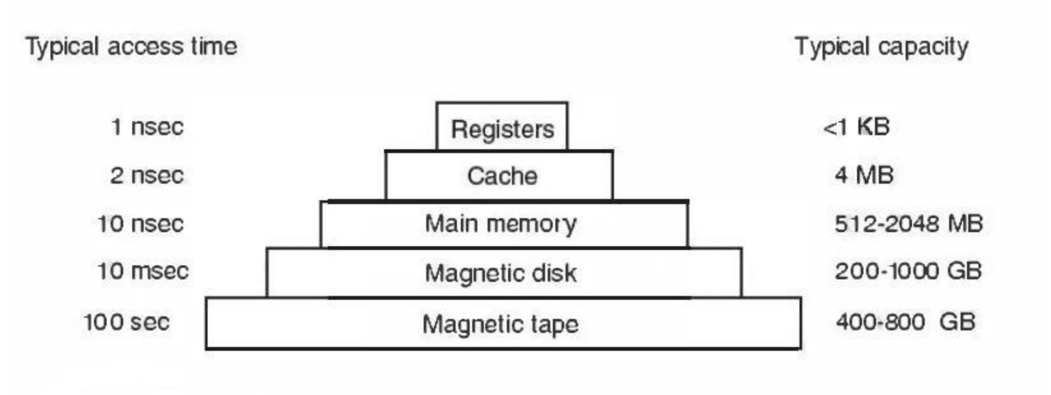


Figure (a) A quad-core chip with a shared L2 cache. (b) A quad-core chip with separate L2 caches.

Memory

- The second major element in any computer is the memory. Preferably, a memory should be very fast (faster than executing an instruction so the CPU is not held up by the memory), abundantly large, and dirt cheap. No current technology satisfies all of these goals, so a different approach is taken. The memory system is built as a hierarchy of layers, as shown in the following figure. The top layers have higher speed, smaller capacity, and greater cost per bit than the lower ones, often by factors of a billion or more.



A typical memory hierarchy. The numbers are very rough approximations.

- The top layer comprises the registers internal to the CPU. They are made of the same material as the CPU and are thus just as fast as the CPU. As a result, there is no delay in accessing them. The storage capacity available in them is typically 32 x 32-bits on a 32-bit CPU and 64 x 64-bits on a 64-bit CPU. Less than 1 KB in both cases. Programs must manage the registers (i.e., decide what to keep in them) themselves, in software.
- Next comes the cache memory, which is generally managed by the hardware. Main memory is divided up into cache lines, typically 64 bytes, with addresses 0 to 63 in cache line 0, addresses 64 to 127 in cache line 1 , and so on. The most heavily used cache lines are kept in a high-speed cache located inside or very close to the CPU. When the program requires to read a memory word, the cache hardware checks to see if the line required is in the cache. If it is, called a cache hit, the request is satisfied from the cache and no memory request is sent over the bus to the main memory. Cache hits generally take about two clock cycles. Cache misses have to go to memory, with a considerable time penalty. Cache memory is limited in size due to its high cost. Some machines have two or even three levels of cache, each one slower and bigger than the one before it.
- Caching plays a main role in various areas of computer science, not just caching lines of RAM. Whenever there is a large resource that can be divided into pieces, some of which are used much more heavily than others, caching is often invoked to get better performance.

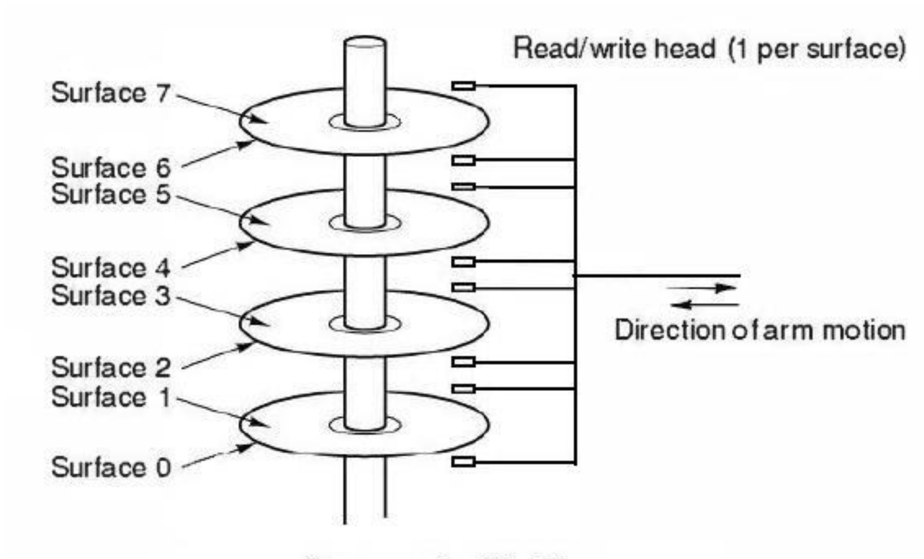
In any caching system, a number of questions come up quite soon, including:

1 When to put a new item into the cache.
2. Which cache line to put the new item in.
3. Which item to remove from the cache when a slot is needed.
4. Where to put a newly evicted item in the larger memory.

- Main memory comes next in the hierarchy of above figure. This is the workhorse of the memory system. Main memory is generally called RAM (Random Access Memory). Old-timers sometimes call it core memory, because computers in the 1950s and 1960s used tiny magnetizable ferrite cores for main memory. At present, memories are hundreds of megabytes to several gigabytes and growing speedily. All CPU requests that cannot be satisfied out of the cache go to main memory.
- In addition to the main memory, a lot of computers have a small amount of nonvolatile random access memory. Unlike RAM, nonvolatile memory does not lose its contents when the power is switched off. ROM (Read Only Memory) is programmed at the factory and cannot be altered afterward. It is fast and inexpensive. On some computers, the bootstrap loader used to start the computer is contained in ROM. Also, some I/0 cards come with ROM for handling low-level device control.
- EEPROM (Electrically Erasable PROM) and flash memory are also nonvolatile, but in contrast to ROM can be removed and rewritten. Nevertheless, writing them takes orders of magnitude more time than writing RAM, so they are used in the same way ROM is, only with the additional feature that it is now possible to correct bugs in programs they hold by rewriting them in the field.
- Flash memory is also frequently used as the storage medium in portable electronic devices. It serves as film in digital cameras and as the disk in portable music players, to name just two uses. Flash memory is intermediate in speed between RAM and disk. Also, unlike disk memory, if it is removed too many times, it wears out.
- Yet another kind of memory is CMOS, which is volatile. Many computers use CMOS memory to hold the current time and date. The CMOS memory and the clock circuit that increases the time in it are powered by a small battery, so the time is correctly updated, even when the computer is unplugged. The CMOS memory can also hold the configuration parameters, such as which disk to boot from. CMOS is used because it draws so little power that the original factory installed battery often lasts for many years. Though, when it begins to fail, the computer can appear to have Alzheimer's disease, forgetting things that it has known for years, like which hard disk to boot from.

Disks

Next in the hierarchy is magnetic disk (hard disk). Disk storage is two orders of magnitude cheaper than RAM per bit and often two orders of magnitude larger as well. The only problem is that the time to randomly access data on it is close to three orders of magnitude slower. This low speed is due to the fact that a disk is a mechanical device, as shown in the following figure.



Structure of a disk drive.

- A disk comprises one or more metal platters that rotate at 5400, 7200, or 10,800 rpm A mechanical arm pivots over the platters from the corner, similar to the pickup arm on an old 33 rpm phonograph for playing vinyl records. Information is written onto the disk in a series of concentric circles. At any given arm position, each of the heads can read an annular region called a track. Together, all the tracks for a given arm position form a cylinder.
- Each track is divided into some number of sectors, typically 512 bytes per sector. On modern disks, the outer cylinders include more sectors than the inner ones. Moving the arm from one cylinder to the next one takes about 1 msec. Moving it to a random cylinder normally takes 5 msec to 10 msec, depending on the drive. Once the arm is on the correct track, the drive must wait for the needed sector to rotate under the head, an additional delay of 5 msec to 10 msec, depending on the drive's rpm. Once the sector is under the head, reading or writing occurs at a rate of 50 MB/sec on low-end disks to 160 MB/sec on faster ones.

Tapes

The last layer in the memory hierarchy is magnetic tape. This medium is frequently used as a backup for disk storage and for holding very large data sets. To access a tape, it must first be put into a tape reader, either by a person or a robot (automated tape handling is common at installations with large databases). Then the tape may have to be spooled forward to get to the requested block. All in all, this could take minutes. The big plus of tape is that it is exceedingly inexpensive per bit and removable, which is important for backup tapes that must be stored off-site in order to survive fires, floods, earthquakes, and other disasters.

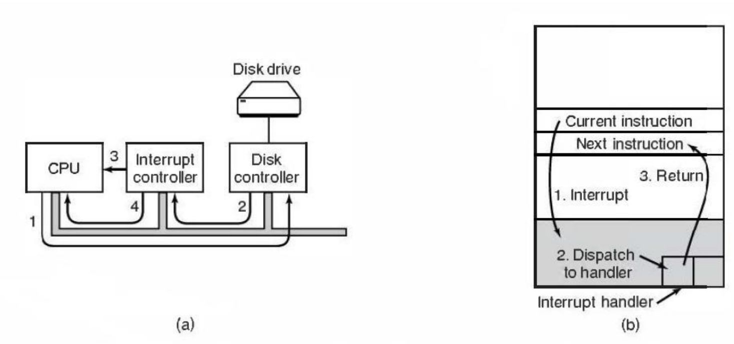
I/O Devices

- The CPU and memory are not the only resources that the operating system must deal with. I/O devices also interact heavily with the operating system. As we saw in "COMPUTER HARDWARE REVIEW" figure, I/O devices are normally made of two parts: a controller and the device itself. The controller is a chip or a set of chips that physically controls the device. It accepts commands from the operating system, for instance, to read data from the device, and carries them out.

- Input and output can be done in three different ways. In the simplest method, a user program issues a system call, which the kernel then translates into a procedure call to the proper driver. The driver then starts the I/O and sits in a tight loop continuously polling the device to see if it is done (generally there is some bit that indicates that the device is still busy). When the I/0 has completed, the driver puts the data (if any) where they are required and returns. The operating system then returns control to the caller. This method is called busy waiting and has the disadvantage of tying up the CPU polling the device until it is ended.

- The second method is for the driver to start the device and ask it to give an interrupt when it is ended. At that point the driver returns. The operating system then blocks the caller if need be and looks for other work to do. When the controller detects the end of the transfer, it creates an interrupt to signal completion.

- The third method for doing I/0 makes use of special hardware: a DMA (Direct Memory Access) chip that can control the flow of bits between memory and some controller without constant CPU interference.

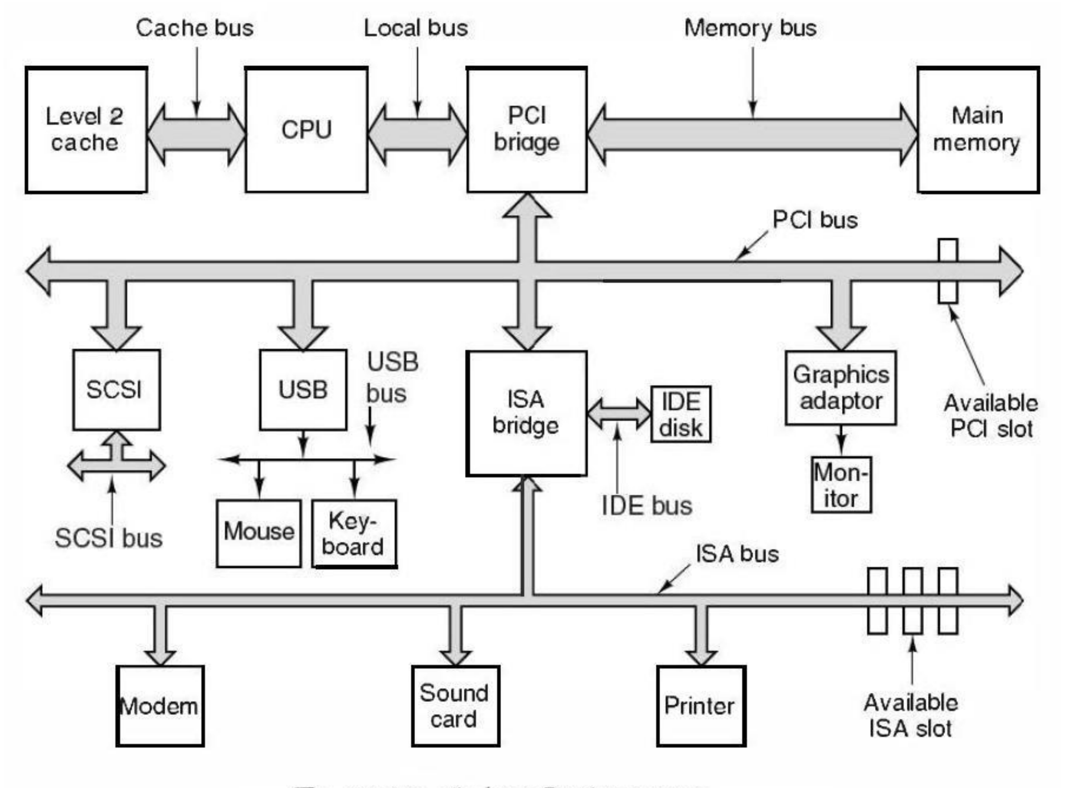


(a) The steps in starting an I/O device and getting an interrupt. (b) Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program.

Buses

The organization of "COMPUTER HARDWARE REVIEW" figure was used on minicomputers for years and also on the original IBM PC. Though, as processors and memories got faster, the ability of a single bus (and certainly the IBM PC bus) to handle all the traffic was constrained to the breaking point. Something had to give. As a result, extra buses were added, both for faster I/O devices and for CPU-to- memory traffic.

As a consequence of this evolution, a large Pentium system currently looks something like the following figure.



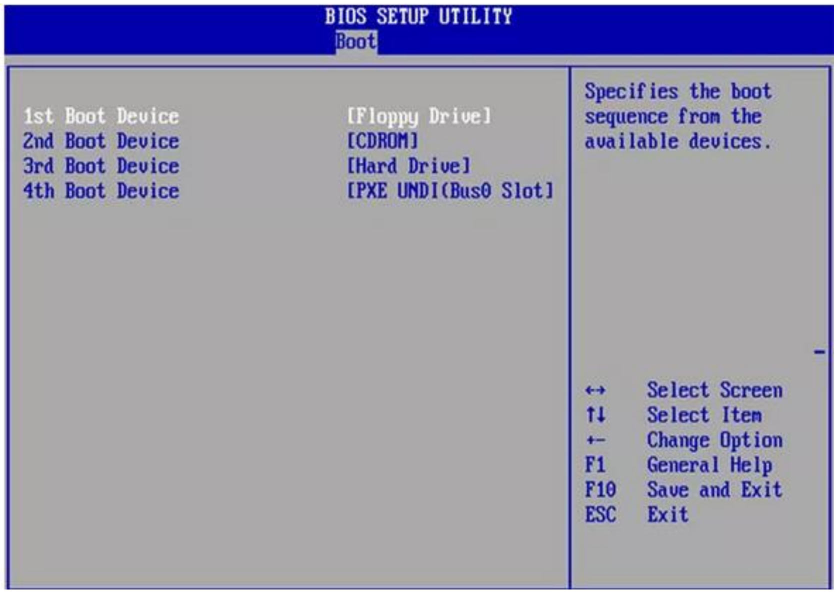
The structure of a large Pentium system

This system has eight buses (cache, local, memory, PCI, SCSI, USB, IDE, and ISA), each with a different transfer rate and function. The operating system must be aware of all of them for configuration and management. The two main buses are the original IBM PC ISA (Industry Standard Architecture) bus and its successor, the PCI (Peripheral Component Interconnect) bus. The ISA bus, which was originally the IBM PC/AT bus, runs at 8.33 MHz and can transfer 2 bytes at once, for a maximum speed of 16.67 MB/sec. It is included for backward compatibility with old and slow I/O cards. Modem systems often leave it out and it is dying off. The PCI bus was invented by Intel as a successor to the ISA bus. It can run at 66 MHz and transfer 8 bytes at a time, for a data rate of 528 MB/sec. Most high-speed I/O devices use the PCI bus now. Even some non-Intel computers use the PCI bus due to the large number of I/O cards available for it.

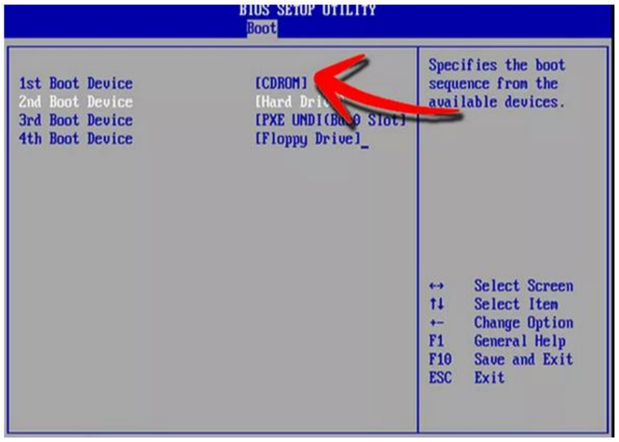
New computers are being brought out with an updated version of the PCI bus called PCI Express.

Booting the Computer

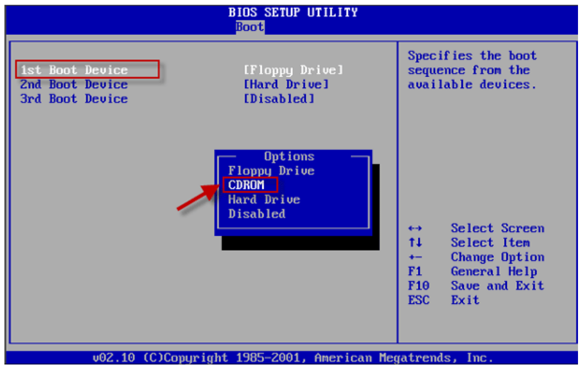
- **Reboot your computer.** Before you can boot from a CD or DVD, you need to make sure that the CD/DVD drive is set as the primary boot device. When you see the manufacturer’s logo, hit the key displayed to enter the BIOS setup menu. The key varies by manufacturer. The most common keys are F2, F10, F12, and Del. The key you need to hit will be displayed underneath the logo, or on the side of the screen.
- Once you are in the BIOS menu, select the Boot submenu. All manufacturers use slightly different BIOS layouts, so look for a variation on the name Boot.



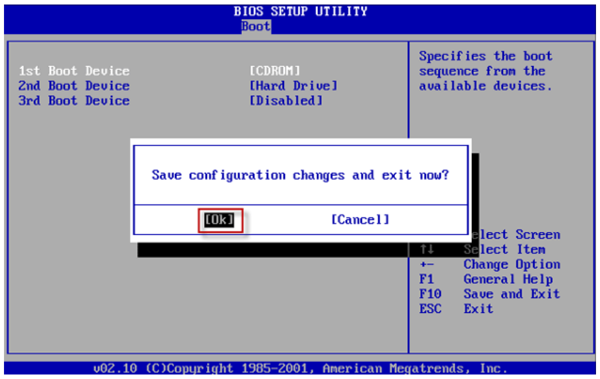
```
                    BIOS SETUP UTILITY
                          Boot

1st Boot Device            [Floppy Drive]          Specifies the boot
2nd Boot Device            [CDROM]                 sequence from the
3rd Boot Device            [Hard Drive]            available devices.
4th Boot Device            [PXE UNDI(Bus0 Slot]

                                                   ↔    Select Screen
                                                   ↑↓   Select Item
                                                   +-   Change Option
                                                   F1   General Help
                                                   F10  Save and Exit
                                                   ESC  Exit
```



```
BIOS SETUP UTILITY
Boot

1st Boot Device          [CDROM]                 Specifies the boot
2nd Boot Device          [Hard Dri               sequence from the
3rd Boot Device          [PXE UNDI(B  0 Slot]    available devices.
4th Boot Device          [Floppy Drive]_

                                                 ↔    Select Screen
                                                 ↑↓   Select Item
                                                 +-   Change Option
                                                 F1   General Help
                                                 F10  Save and Exit
                                                 ESC  Exit
```



```
BIOS SETUP UTILITY
Boot

1st Boot Device          [Floppy Drive]          Specifies the boot
2nd Boot Device          [Hard Drive]            sequence from the
3rd Boot Device          [Disabled]              available devices.

                         Options
                         Floppy Drive
                         CDROM
                         Hard Drive
                         Disabled
                                                 ↔    Select Screen
                                                 ↑↓   Select Item
                                                 +-   Change Option
                                                 F1   General Help
                                                 F10  Save and Exit
                                                 ESC  Exit

v02.10 (C)Copyright 1985-2001, American Megatrends, Inc.
```



```
BIOS SETUP UTILITY
Boot

1st Boot Device          [CDROM]                 Specifies the boot
2nd Boot Device          [Hard Drive]            sequence from the
3rd Boot Device          [Disabled]              available devices.

        Save configuration changes and exit now?

             [Ok]                [Cancel]
                                                      lect Screen
                                                 ↑↓   Select Item
                                                 +-   Change Option
                                                 F1   General Help
                                                 F10  Save and Exit
                                                 ESC  Exit

v02.10 (C)Copyright 1985-2001, American Megatrends, Inc.
```

<u>THE OPERATING SYSTEM ZOO</u>

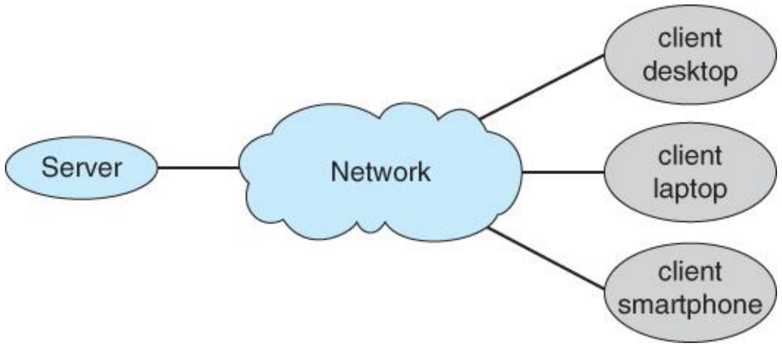
<u>Mainframe Operating Systems</u>

- room-sized computers
- mainframe is with 1000 disks and millions of gigabytes of data
- Mainframes are also making something of a comeback as high-end Web servers, servers for large-scale electronic commerce sites, and servers for business-to-business transactions.

- They typically offer three kinds of services: batch, transaction processing, and timesharing.

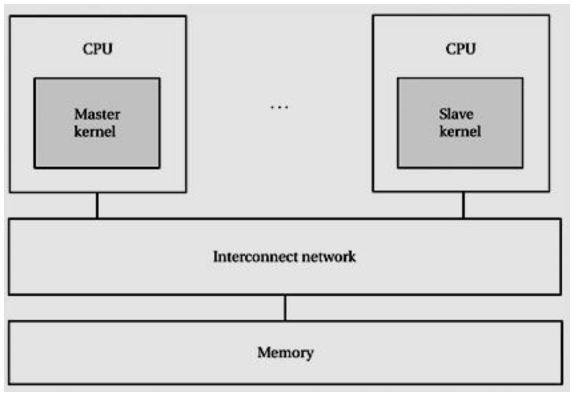
Server Operating Systems

- They run on servers, which are either very large personal computers, workstations, or even mainframes.
- They serve multiple users at once over a network and allow the users to share hardware and software resources.



Multiprocessor Operating Systems

- Connect multiple CPUs into a single system.
- they are connected and shared, these systems are called parallel computers, multicomputer, or multiprocessors.



Personal Computer Operating Systems

- A **personal computer** (**PC**) is a general- purpose computer whose size, capabilities, and original sale price make it useful for individuals, and is intended to be operated directly by an end-user with no intervening computer time-sharing models that allowed larger, more expensive minicomputer and mainframe systems to be used by many people, usually at the same time.

Handheld Computer Operating Systems

- A handheld computer, originally known as a **PDA** (**Personal Digital Assistant**), is a small computer that can be held in your hand during operation. Smartphones and tablets are the best-known examples.

Embedded Operating Systems

- Embedded systems run on the computers that control devices that are not generally thought of as computers and which do not accept user-installed software.
- Typical examples are microwave ovens, TV sets, cars, DVD recorders, traditional phones, and MP3 players.

Sensor-Node Operating Systems

- Networks of tiny sensor nodes are being deployed for numerous purposes.
- These nodes are tiny computers that communicate with each other and with a base station using wireless communication. Sensor networks are used to protect the perimeters of buildings, guard national borders, detect fires in forests, measure temperature and precipitation for weather forecasting, glean information about enemy movements on battlefields, and much more.

Real-Time Operating Systems

- If the action absolutely *must* occur at a certain moment (or within a certain range), we have a **hard real-time system**. Many of these are found in industrial process control, avionics, military and similar application areas. These systems must provide absolute guarantees that a certain action will occur by a certain time.
- A **soft real-time system**, is one where missing an occasional deadline, while not desirable, is acceptable and does not cause any permanent damage. Digital audio or multimedia systems fall in this category. Smartphones are also soft real time systems.

Smart Card Operating Systems

- The smallest operating systems run on smart cards, which are credit-card-sized devices containing a CPU chip. They have very severe processing power and memory constraints.

OPERATING SYSTEM CONCEPTS

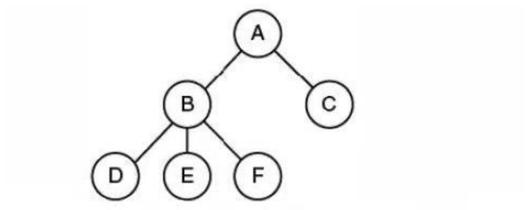
PROCESSES

- Processes-- A process is basically a program in execution. The execution of a process must progress in a sequential fashion.
- A process is defined as an entity which represents the basic unit of work to be implemented in the system. To put it in simple terms, we write our computer

programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

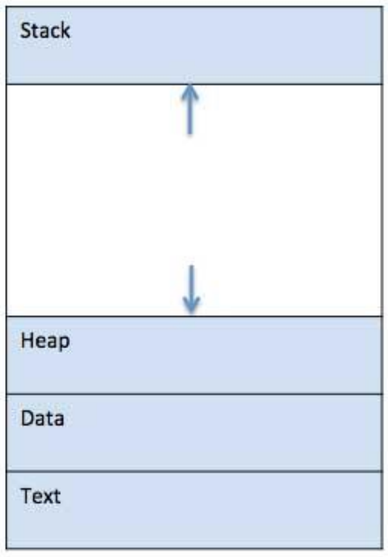
- When a program is loaded into the memory and it becomes a process, it can be divided into four sections – stack, heap, text and data. The following image shows a simplified layout of a process inside main memory −

If a process can create one or more other processes (referred to as child processes) and these processes in turn can produce child processes, we quickly arrive at the process tree structure of the following figure. Related processes that are cooperating to get some job done frequently need to communicate with one another and synchronize their activities. This communication is called interprocess communication, and will be addressed in detail in "PROCESSES AND THREADS".



A process tree. Process *A* created two child processes, *B* and *C*. Process *B* created three child processes, *D*, *E*, and *F*.

Other process system calls are available to request more memory (or release unused memory), wait for a child process to come to an end, and overlay its program with a different one.



S.N.	Component & Description
1	**Stack** **The process Stack contains the temporary data such as method/function parameters, return address and local variables.**
2	**Heap** **This is dynamically allocated memory to a process during its run time.**
3	**Text** **This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.**
4	**Data** **This section contains the global and static variables.**

Address space

- Address space is the amount of memory allocated for all possible addresses for a computational entity, such as a device, a file, a server, or a networked computer. Address space may refer to a range of either physical or virtual addresses accessible to a processor or reserved for a process.
- As unique identifiers of single entities, each address specifies an entity's *location* (unit of memory that can be addressed separately). On a computer, each computer device and process is allocated address space, which is some portion of the processor's address space.

On the other hand, on many computers addresses are 32 or 64 bits, giving an address space of 2^{32} or 2^{64} bytes, respectively. What happens if a process has more address space than the computer has main memory and the process wants to use it all?
These days, a technique called virtual memory exists, as mentioned earlier, in which the operating system keeps part of the address space in main memory and part on disk and shuttles pieces back and forth between them as required. In essence, the operating system creates the abstraction of an address space as the set of addresses a process may reference. The address space is decoupled from the machine's physical memory, and may be either larger or smaller than the physical memory.

File

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

<u>File Structure</u>

- A File Structure should be according to a required format that the operating system can understand.
- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

<u>File Type</u>

- File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files –'

1. **Ordinary files**

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

2. Directory files

- These files contain list of file names and other information related to these files.

3. Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

- These files are of two types −
- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

- File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −
- Sequential access
- Direct/Random access
- Indexed sequential access

1. **Sequential access**

- A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

2. Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

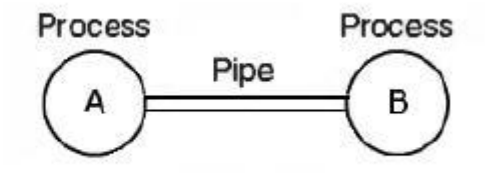
3. Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Input/Output

- All computers have physical devices for acquiring input and producing output.
- Many kinds of input and output devices exist, including keyboards, monitors, printers, and so on. It is up to the operating system to manage these devices.
- Consequently, every operating system has an I/O subsystem for managing its I/O devices. Some of the I/O software is device independent, that is, applies to many or all I/O devices equally well. Other parts of it, such as device drivers, are specific to particular I/O devices

All computers have physical devices for obtaining input and generating output. After all, what good would a computer be if the users could not tell it what to do and could not get the results after it did the work requested? Many types of input and output devices exist, including keyboards, monitors, printers, and so on. It is up to the operating system to manage these devices. As a result, every operating system has an I/O subsystem for managing its I/O devices. Some of the I/O software is device independent, that is, applies to many or all I/O devices equally well. Other parts of it, such as device drivers, are specific to particular I/O devices. In "INPUT/OUTPUT" we will have a look at I/O software.



Two processes connected by a pipe

Security

- Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. We're going to discuss following topics in this chapter.
- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Authentication

- Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −
- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.

- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords

- One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.
- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Program Threats

- Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.
- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

System Threats

- System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.
- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

Unix - What is Shells?

- The shell provides you with an interface to the UNIX system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.
- A shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of shells, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell Prompt:

- The prompt, $, which is called command prompt, is issued by the shell. While the prompt is displayed, you can type a command.
- The shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.
- Following is a simple example of **date** command which displays current date and time:
- $date Thu Jun 25 08:30:19 MST 2009

Shell Types:

- In UNIX there are two major types of shells:
- The Bourne shell. If you are using a Bourne-type shell, the default prompt is the $ character.
- The C shell. If you are using a C-type shell, the default prompt is the % character.

- There are again various subcategories for Bourne Shell which are listed as follows:
- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

The different C-type shells follow:

- C shell (csh)
- TENEX/TOPS C shell (tcsh)

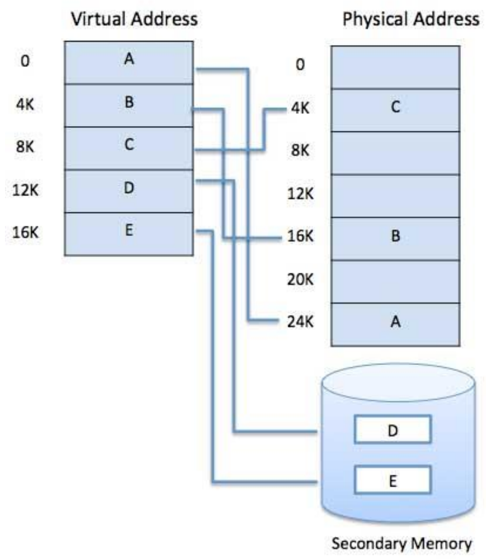
- The original UNIX shell was written in the mid-1970s by Stephen R. Bourne while he was at AT&T Bell Labs in New Jersey.
- The Bourne shell was the first shell to appear on UNIX systems, thus it is referred to as "the shell".
- The Bourne shell is usually installed as /bin/sh on most versions of UNIX. For this reason, it is the shell of choice for writing scripts to use on several different versions of UNIX.

Operating System - Virtual Memory

- A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.
- The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

- Following are the situations, when entire program is not required to be loaded fully in main memory.
- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.

- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

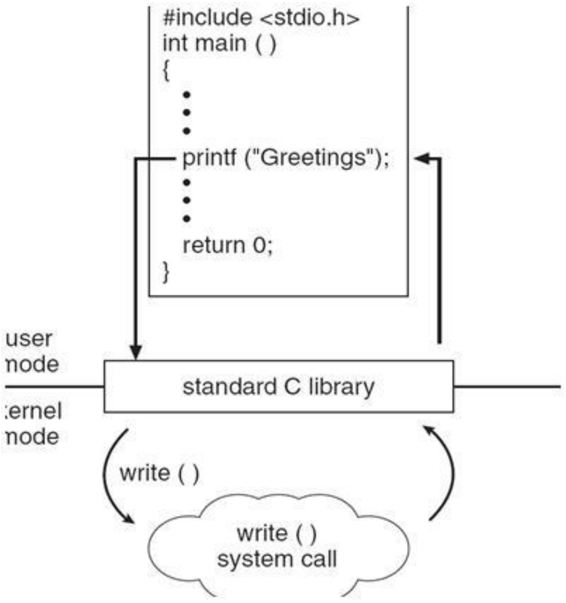


System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API)
- Three most common APIs are
  - Win32 API for Windows,
  - POSIX API (all versions of UNIX, Linux, and Mac OS X), and
  - Java API for the Java virtual machine (JVM)

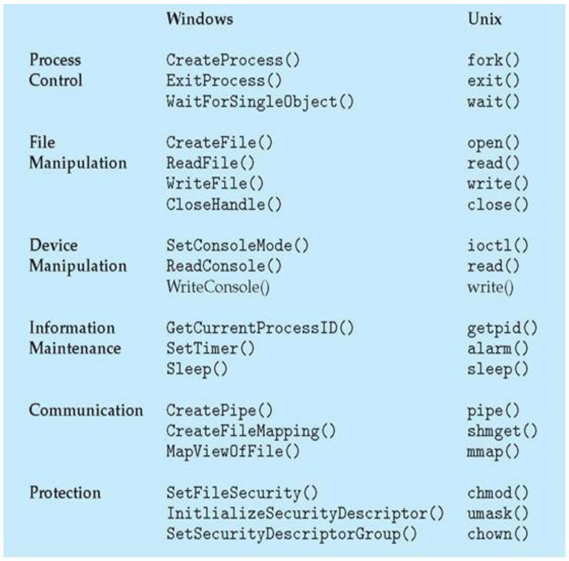
Example

- The standard C library provides a portion of the system call interface for many version of UNIX and Linux.
- As an example, let's assume a C program invokes printf() statement.
- The C library intercepts this call and invokes the necessary system call(s) in the OS.
- The C library takes the value returned by write() and passes it back to the user program



```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```

Types



	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

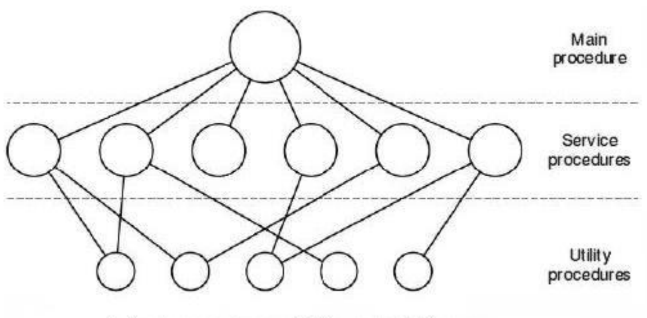
OPERATING SYSTEM STRUCTURE

Monolithic Systems

Undoubtedly the most common organization, in this approach the entire operating system runs as a single program in kernel mode. The operating system is written as a collection of procedures, linked together into a single large executable binary program. When this technique is used, each procedure in the system is free to call any other one, if the latter provides some useful computation that the former needs. Having thousands of procedures that can call each other without restriction often leads to an unwieldy and difficult to understand system. This organization suggests a basic structure for the operating system:

1. A main program that invokes the requested service procedure.
2. A set of service procedures that carry out the system calls.
3. A set of utility procedures that help the service procedures.

In this model, for each system call there is one service procedure that takes care of it and executes it. The utility procedures do things that are required by various service procedures, such as fetching data from user programs. This division of the procedures into three layers is shown in the following figure.



A simple structuring model for a monolithic system.

Layered Systems

The system had six layers, as shown in the following figure. Layer 0 dealt with allocation of the processor, switching between processes when interrupts occurred or timers expired.
Layer 1 did the memory management. It distributed space for processes in main memory and on a 512K word drum used for holding parts of processes (pages) for which there was no room in main memory.
Layer 2 handled communication between each process and the operator console (that is, the user).

Microkernels

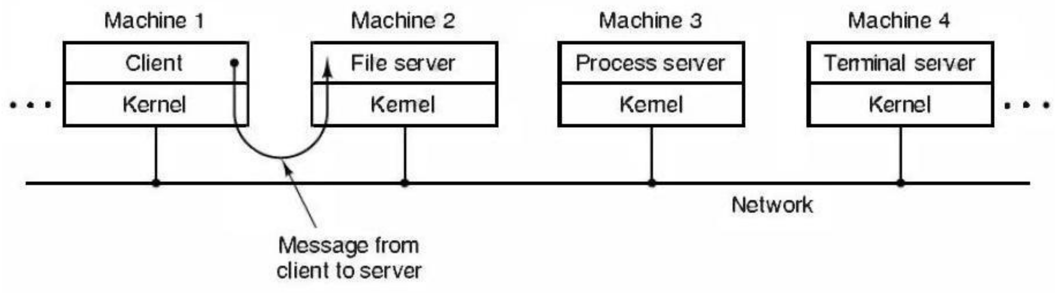
The main idea behind the microkernel design is to attain high reliability by splitting the operating system up into small, well-defined modules, only one of which-the microkernel-runs in kernel mode and the rest run as relatively powerless normal user processes. Particularly, by running each device driver and file system as a separate user process, a bug in one of these can crash that component, but cannot crash the whole system. Thus a bug in the audio driver will cause the sound to be garbled or stop, but will not crash the computer. On the contrary, in a monolithic system with all the drivers in the kernel, a buggy audio driver can easily reference an invalid memory address and bring the system to a grinding halt immediately.

Client-Server Model / Virtual Machines

Client-Server Model

A small variation of the microkernel idea is to differentiate two classes of processes, the servers, each of which gives some service, and the clients, which use these services. This model is known as the client-server model. Sometimes the lowest layer is a microkernel, but that is not required. The essence is the presence of client processes and server processes.

Communication between clients and servers is often by message passing. To get a service, a client process constructs a message saying what it wants and sends it to the appropriate service. The service then does the work and sends back the answer.



The client-server model over a network.

Virtual Machines

In computing, a **virtual machine** (**VM**) is an emulation of a given computer system. **Virtual machines** operate based on the computer architecture and functions of a real or hypothetical computer, and their implementations may involve specialized hardware, software, or a combination.

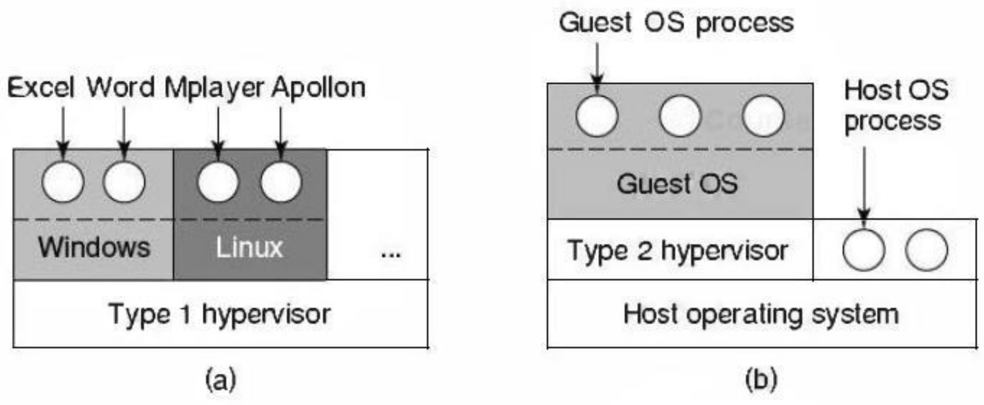


Figure 1. (a) A type I hypervisor. (b) A type 2 hypervisor.

The Java Virtual Machine

One more area where virtual machines are used, but in a somewhat different way, is for running Java programs. When Sun Microsystems invented the Java programming language, it also invented a virtual machine (i.e., a computer architecture) called the JVM (Java Virtual Machine). The Java compiler produces code for JVM, which then normally is executed by a software JVM interpreter. The advantage of this technique is that the JVM code can be shipped over the Internet to any computer that has a JVM interpreter and run there.

Exokernels

Instead of cloning the actual machine, as is done with virtual machines, another approach is partitioning it, in other words, giving each user a subset of the resources. In this way one virtual machine might get disk blocks 0 to 1023, the next one might get blocks 1024 to 2047, and so on.

The advantage of the exokernel scheme is that it saves a layer of mapping. In the other designs, each virtual machine thinks it has its own disk, with blocks running from 0 to some maximum, so the virtual machine monitor must maintain tables to remap disk addresses (and all other resources). With the exokernel, this remapping is not required. The exokernel require only keep track of which virtual machine has been assigned which resource. This technique still has the advantage of separating the multiprogramming (in the exokernel) from the user operating system code (in user space), but with less overhead, since all the exokernel has to do is keep the virtual machines out of each other's hair.