UNIT -1

Introduction

An *operating system* acts as an intermediary between the user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a *convenient* and *efficient* manner.

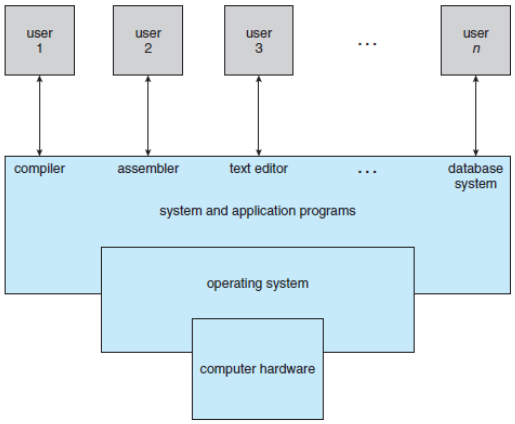
An **operating system** is a program that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. Operating systems are designed to be ***convenient,*** others to be ***efficient,*** and others to be some combination of the two.

COMPUTER SYSTEM ORGANIZATION

A computer system can be divided roughly into four components:

- the ***hardware,***
- the ***operating system,***
- the ***application programs,*** and
- the ***users***

The **hardware**—the **central processing unit (CPU)**, the **memory**, and the **input/output (I/O) devices**—provides the basic computing resources for the system. The **application programs**—such as word processors, spreadsheets, compilers, and Web browsers— define the ways in which these resources are used to solve users' computing problems. The operating system controls the hardware and coordinates its use among the various application programs for the various users.



User View

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

In this case, the operating system is designed mostly for **ease of use**, with some attention paid to performance and none paid to **resource utilization**—how various hardware and software resources are shared.

Performance is, of course, important to the user; but such systems are optimized for the single-user experiencerather than the requirements of multiple users.

System View

From the computer's point of view, the operating system is the program most intimately involvedwith the hardware.

In this context, we can view an operating system as a **resource allocator**.

A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources. The operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly.

Basic elements

A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory (Figure 1.2).

Each devicecontroller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays).

For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run. This initial program, or **bootstrap program**, tends to be simple.

Typically, it is stored within the computer hardware in read-only memory (**ROM**) or electrically erasable programmable read-only memory (**EEPROM**), known by the general term **firmware**. It initializes all aspects of the system, from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the operating system and how to start executing that system. To accomplish this goal, the bootstrap program mustlocate the operating-system kernel and load it into memory. Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**, or **system daemons** that run the entire time the kernel is running.

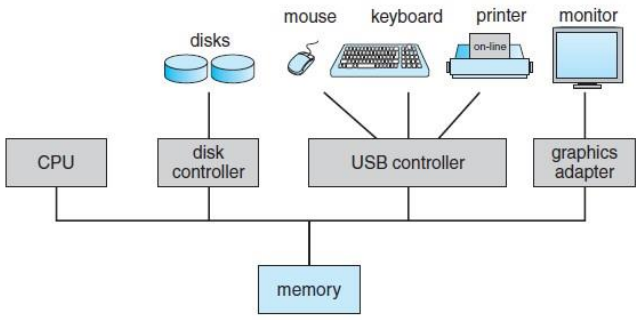


Figure 1.2 A modern computer system.

Instruction Execution and Interrupts

The occurrence of an event is usually signaled by an **interrupt** from either the hardware or the software. Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the

system bus. Software may trigger an interrupt by executing a special operation called a **system call** (also called a **monitor call**).

When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located. The interrupt service routine executes; on completion, the CPU resumes the Interrupted computation. A timeline of this operation is shown in Figure 1.3.

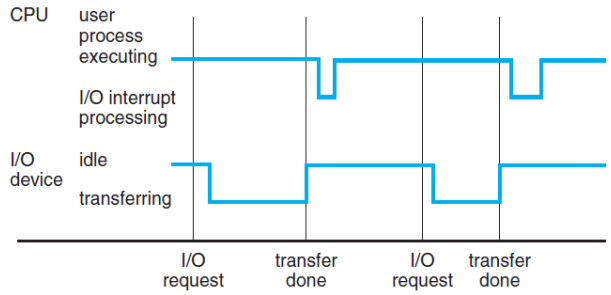


Figure 1.3 Interrupt timeline for a single process doing output.

Memory Hierarchy

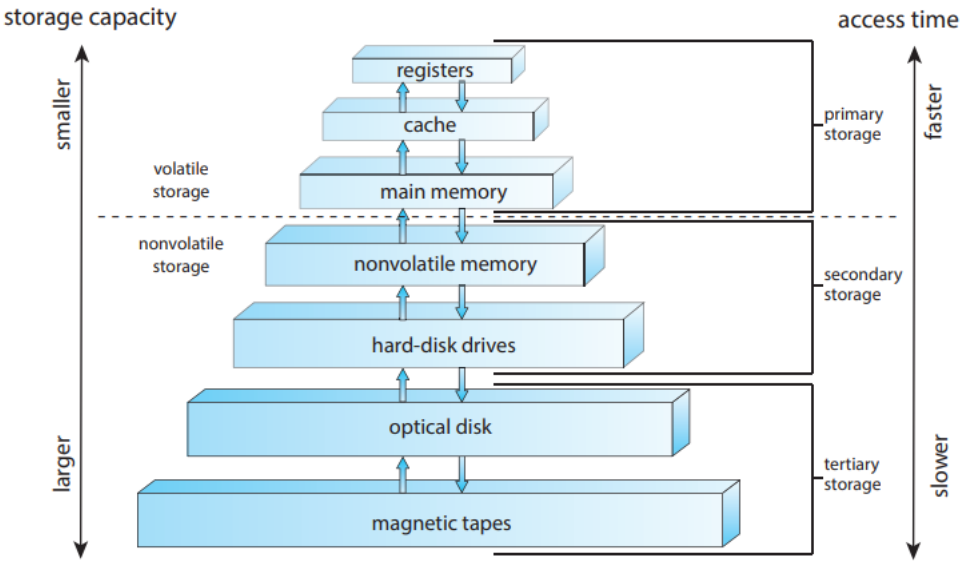


Figure 1.6 Storage-device hierarchy.

The wide variety of storage systems can be organized in a hierarchy (Figure 1.4) according to speed and cost. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases. The top four levels of memory in Figure 1.4 may be constructed using semiconductor memory.

In addition to differing in speed and cost, the various storage systems are either volatile or nonvolatile. As mentioned earlier, **volatile storage** loses its contents when the power to the device isremoved. In the absence of expensive battery and generator backup systems, data must be written to **nonvolatile storage** for safekeeping. In the hierarchy shown in Figure 1.4, the storage systems above the solid-state disk are volatile, whereas those including the solid-state disk and below are nonvolatile.

Solid-state disks have several variants but in general are faster than magnetic disks and are nonvolatile. One type of solid-state disk stores data in a large DRAM array during normal operation but also contains a hidden magnetic hard disk and a battery for backup power. If external power is interrupted, this solid-state disk's controller copies the data from RAM to the magnetic disk. When external power is restored, the controller copies the data back into RAM. Another form of solid-state disk is flash memory, which is popular in cameras and **personal digital assistants (PDAs)**, in robots, and increasingly for storage on general-purpose computers. Flash memory is slower than DRAM but needs no power to retain its contents. Another form of nonvolatile storage is **NVRAM**, which is DRAM with battery backup power. This memory can be as fast as DRAM and (as long as the battery lasts) is nonvolatile.

Cache Memory

- 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

Direct Memory Access

Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O. To solve this problem, **direct memory access (DMA)** is used. After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only oneinterrupt is generated per block, to tell the device driver that the operation has completed, rather than the one interrupt per byte generated for low-speed devices. While the device controller is performing these operations,the CPU is available to accomplish other work.

Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rather than competing for cycles on a shared bus. In this case, DMA is even more effective. Figure 1.5 shows the interplay of all components of a computer system.

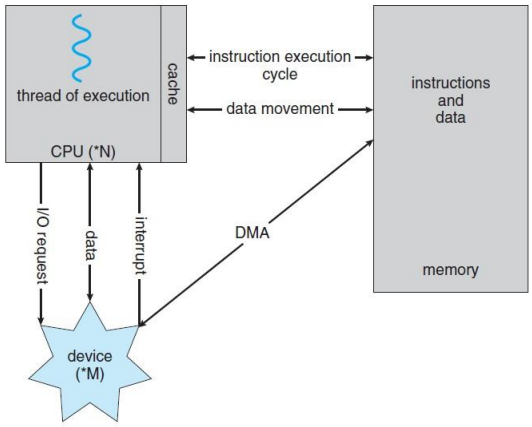


Figure 1.5 How a modern computer system works.

COMPUTER-SYSTEM ARCHITECTURE

Multiprocessor Organization

Multiprocessor systems (also known as **parallel systems** or **multicore systems**) have begun to dominate the landscape of computing. Such systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices. Recently, multiple processors have appeared on mobile devices such as smart phones and tablet computers.

Multiprocessor systems have three main advantages:

1. Increased throughput. By increasing the number of processors, we expect to get more work done in less time.

2. Economy of scale. Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. If several programs operate on the same set of data, it is cheaper to store those data on one disk and to have all the processors share them than to have many computers with local disks and many copies of the data.

3. Increased reliability. If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

The multiple-processor systems in use today are of two types. Some systems use **asymmetricmultiprocessing**, in which each processor is assigned a specific task.

A ***boss*** processor controls the system; theother processors either look to the boss for instruction or have predefined tasks. This scheme defines a boss–worker relationship. The boss processor schedules and allocates work to the worker processors. The most common systems use **symmetric multiprocessing (SMP)**, in which each processor performs all tasks within the operating system. SMP means that all processors are peers; no boss–worker relationship exists between processors. Figure 1.6 illustrates a typical SMP architecture.

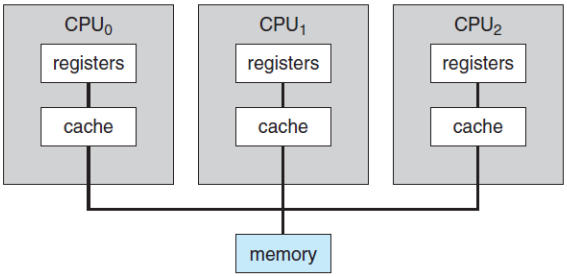


Figure 1.6 Symmetric multiprocessing architecture.

Multiprocessing can cause a system to change its memory access model from uniform memory access (**UMA**)to non-uniform memory access (**NUMA**). UMA is defined as the situation in which access to any RAM from any CPU takes the same amount of time. With NUMA, some parts of memory may take longer to access than other parts, creating a performance penalty. Operating systems can minimize the NUMA penalty through resource management

Multicore Organization

A more recent, similar trend in system design is to pa multiple computing cores on a single chip. Each core appears as a separate processor to the operating system. Whether the cores appear across CPU chips or within CPU chips, we call these systems **multicore** or **multiprocessor** systems. Multithreaded programming provides a mechanism for more efficient use of these multiple computing cores and improved concurrency.

Consider an application with four threads. On a system with a single computing core, concurrency merely means that the execution of the threads will be interleaved over time (Figure 4.3), because the processing core is capable of executing only one thread at a time. On a system with multiple cores, however, concurrency means that the threads can run inparallel, because the system can assign a separate thread to each core (Figure 4.4).

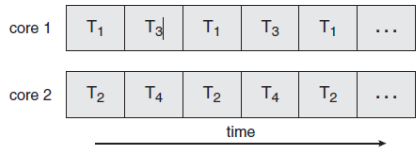


Figure 4.4 Parallel execution on a multicore system.

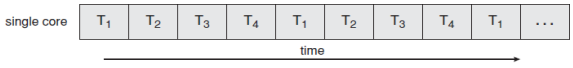
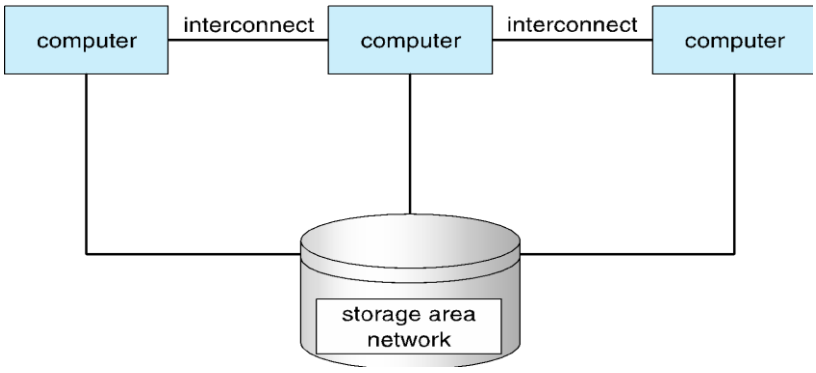


Figure 4.3 Concurrent execution on a single-core system.

Distinction between ***parallelism*** and ***concurrency***. A system is parallel if it can perform more than one task simultaneously. In contrast, a concurrent system supports more than one task by allowing all the tasks to make progress. Thus, it is possible to have concurrency without parallelism.

Clustered Systems

Another type of multiprocessor system is a clustered system, which gathers multiple CPUs. Clustered systems differ from the multiprocessor systems in that they are composed of two or more individual systems—or nodes—joined together; each node is typically a multicore system. Such systems are considered loosely coupled. Clustered computers share storage and are closely linked via a local-area network LAN or a faster interconnect, such as InfiniBand.



General structure of a clustered system.

- **Clustering** is usually used to provide high-availability service— that is, service that will continue even if one or more systems in the cluster fail.
- Generally, we obtain high availability by adding a level of redundancy in the system. A layer of cluster software runs on the cluster nodes. Each node can monitor one or more of the others (over the network). If the monitored machine fails, the monitoring machine can take ownership of its storage and restart the applications that were running on the failed machine.
- The users and clients of the applications see only a brief interruption of service. High availability provides increased reliability, which is crucial in many applications. The ability to continue providing service proportional to the level of surviving hardware is called **graceful degradation**.
- Some systems go beyond graceful degradation and are called fault tolerant, because they can suffer a failure of any single component and continue operation. Fault tolerance requires a mechanism to allow the failure to be detected, diagnosed, and, if possible, corrected.
- Clustering can be structured **asymmetrically or symmetrically**. In **asymmetric clustering**, one machine is in hot-standby mode while the other is running the applications. The hot-standby host machine does nothing but monitor the active server. If that server fails, the hot-standby host becomes the active server.
- In **symmetric clustering**, two or more hosts are running applications and are monitoring each other. This structure is obviously more efficient, as it uses all of the available hardware. However, it does require that more than one application be available to run.

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

- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When it has to wait (for I/O for example), OS switches to another job

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

OPERATING-SYSTEM OPERATIONS

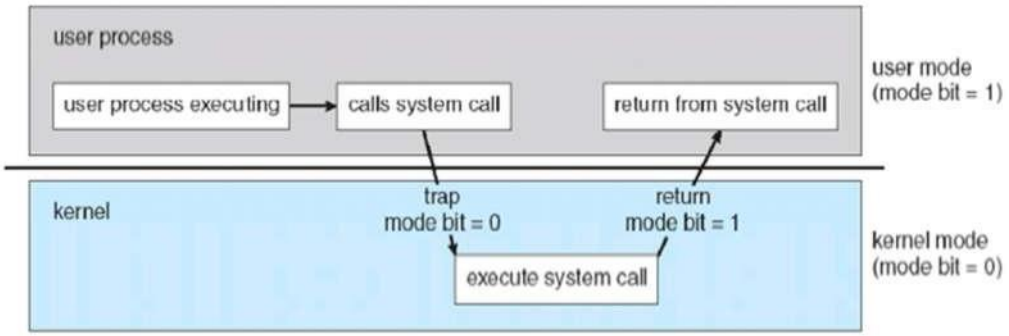
- If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap.
- A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed. The interrupt-driven nature of an operating system defines that system's general structure.
- Without protection against these sorts of errors, either the computer must execute only one process at a time or all output must be suspect. A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

Multiprogramming and Multi-Tasking

- **Multiprogramming** increases CPU utilization, as well as keeping users satisfied, by organizing programs so that the CPU always has one to execute. In a multiprogrammed system, a program in execution is termed a process
- **Multitasking** is a logical extension of multiprogramming. In multitasking systems, the CPU executes multiple processes by switching among them, but the switches occur frequently, providing the user with a fast response time.

Dual-Mode and Multimode Operation

- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.



- At the very least, we need two separate modes of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode).
- A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). With the mode bit, we can distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.

Timer

- We cannot allow a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system.
- To accomplish this goal, we can use a timer. A timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second).
- A variable timer is generally implemented by a fixed-rate clock and a counter.
- The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs. For instance, a 10-bit counter with a 1-millisecond clock allows interrupts at intervals from 1 millisecond to 1,024 milliseconds, in steps of 1 millisecond.

BASIC FUNCTIONS OF OPERATION SYSTEM:

The various functions of operating system are as follows:

1. Process Management:

• A program does nothing unless their instructions are executed by a CPU.A process is a program in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual user on a pc is a process.

• A system task such as sending output to a printer is also a process. A process needs certain resources including CPU time, memory files & I/O devices to accomplish its task.

• These resources are either given to the process when it is created or allocated to it while it is running. The OS is responsible for the following activities of process management.

• Creating & deleting both user & system processes.

• Suspending & resuming processes.

• Providing mechanism for process synchronization.

• Providing mechanism for process communication.

• Providing mechanism for deadlock handling.

2. Main Memory Management:

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared

by the CPU & I/O device. The central processor reads instruction from main memory during instruction fetch cycle & it both reads &writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address & access directly. For example, for the CPU to process data from disk. Those data must first be transferred to main memory by CPU generated E/O calls. Instruction must be in memory for the CPU to execute them. The OS is responsible for the following activities in connection with memory management.

• Keeping track of which parts of memory are currently being used & by whom.
• Deciding which processes are to be loaded into memory when memory space becomes available.
• Allocating &deallocating memory space as needed.

3. File Management:

File management is one of the most important components of an OS computer can store information on several different types of physical media magnetic tape, magnetic disk & optical disk are the most common media. Each medium is controlled by a device such as disk drive or tape drive those has unique characteristics. These characteristics include access speed, capacity, data transfer rate & access method (sequential or random).For convenient use of computer system the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. A file is collection of related information defined by its creator. The OS is responsible for the following activities of file management.

• Creating & deleting files.
• Creating & deleting directories.
• Supporting primitives for manipulating files & directories.
• Mapping files into secondary storage.
• Backing up files on non-volatile media.

4. I/O System Management:

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem. The I/O subsystem consists of:

• A memory management component that includes buffering, catching & spooling.
• A general device- driver interfaces drivers for specific hardware devices. Only the device driver knows the peculiarities of the specific device to which it is assigned.

5. Secondary Storage Management:

The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. The operating system is responsible for the following activities of disk management.

• Free space management.
• Storage allocation.
• Disk scheduling

Because secondary storage is used frequently it must be used efficiently.

6.Protection or security:

If a computer system has multi users & allow the concurrent execution of multiple processes then the various processes must be protected from one another's activities. For that purpose, mechanisms ensure that files, memory segments, CPU & other resources can be operated on by only those processes that have gained proper authorization from the OS.

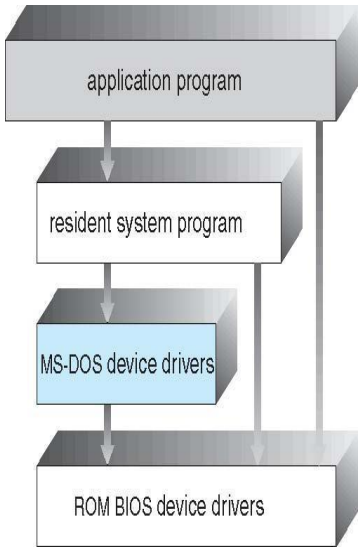
OPERATING SYSTEM STRUCTURE

Various ways to structure OS

- Simple structure – MS-DOS
- More complex -- UNIX
- Layered – an abstrcation
- Microkernel -Mach

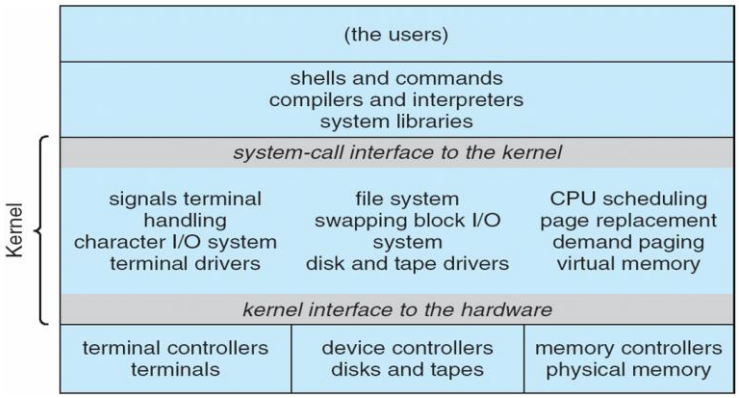
MS-DOS System Structure

- MS-DOS – written to provide the most functionality in the least space.
- Not divided into modules.
- Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.



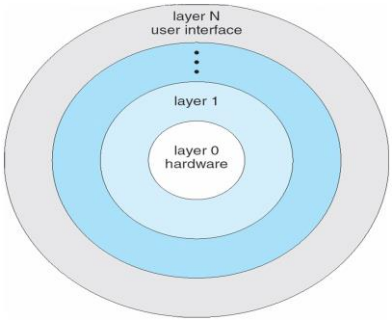
Unix System Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts.
- Systems programs – use kernel supported system calls to provide useful functions such as compilation and file manipulation.
- The kernel - Consists of everything below the system-call interface and above the physical hardware
- Provides the file system, CPU scheduling, memory management, and other operating- system functions; a large number of functions for one level.



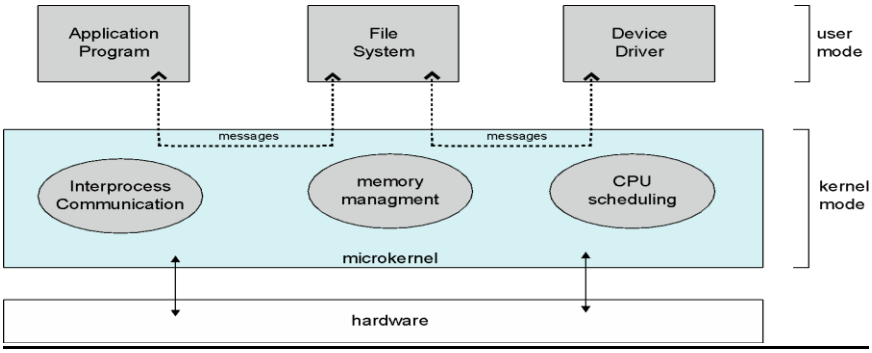
Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- An OS layer is an implementation of an abstract object that is the encapsulation of data and operations that can manipulate those data. These operations (routines) can be invoked by higher-level layers. The layer itself can invoke operations on lower-level layers.
- Layered approach provides modularity. With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers.
- Each layer is implemented by using only those operations that are provided lower level layers.
- The major difficulty is appropriate definition of various layers.



Microkernel System Structure

- Moves as much from the kernel into "user" space.
- Communication takes place between user modules using message passing.
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure



SYSTEM CALLS

- System calls provide the interface between a process and the operating system.
- These calls are generally available as assembly-language instructions.
- System calls can be grouped roughly into five major categories:
  1. Process control
  2. file management
  3. device management
  4. information maintenance
  5. communications.

1. Process Control
   - end,abort
   - load, execute
   - Create process and terminate process
   - get process attributes and set process attributes.
   - wait for time, wait event, signal event
   - Allocate and free memory.
2. File Management
   - Create file, delete file
   - Open , close
   - Read, write, reposition
   - Get file attributes, set file attributes.
3. Device Management
   - Request device, release device.
   - Read, write, reposition
   - Get device attribtues, set device attributes
   - Logically attach or detach devices
4. Information maintenance
   - Get time or date, set time or date
   - Get system data, set system data
   - Get process, file, or device attributes
   - Set process, file or device attributes
5. Communications
   - Create, delete communication connection
   - Send, receive messages
   - Transfer status information
   - Attach or detach remote devices

Two types of communication models
(a) Message passing model
(b) Shared memory model

SYSTEM PROGRAMS

- System programs provide a convenient environment for program development and execution.
- They can be divided into several categories:

1. File management: These programs create, delete, copy, rename, print, dump, list and generally manipulate files and directories.
2. Status information: The status such as date, time, amount of available memory or diskspace, number of users or similar status information.
3. File modification: Several text editors may be available to create and modify the content of files stored on disk or tape.
4. Programming-language support: Compilers, assemblers, and interpreters for common programming languages are often provided to the user with the operating system.
5. Program loading and execution: The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders.
6. Communications: These programs provide the mechanism for creating virtual connections among processes, users, and different computer systems. (email, FTP, Remote log in)
7. Application programs: Programs that are useful to solve common problems, or to perform common operations. Eg. Web browsers, database systems.

OS Generation and System Boot.
OS Generation

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.

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.

Another feature present in this generation was time-sharing technique, a variant of multiprogrammingtechnique, in which each user has an on-line (i.e., directly connected) terminalTimesharing systems were developed to multiprogram large number of simultaneous interactive users.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in thesystem 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.

System Boot.

- Operating system must be made available to hardware so hardware can start it

- Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
- Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
- When power initialized on system, execution starts at a fixed memory location
    - Firmware used to hold initial boot code