LECTURE NOTES

UNIT-I

Syllabus: Introduction: Concept of Operating Systems, Generations of Operating systems, Types of Operating Systems, OS Services, System Calls, Structure of an OS - Layered, Monolithic, Microkernel Operating Systems, Concept of Virtual Machine.

Introduction :

What is an Operating System?

- An operating system is a program that manages the computer hardware.
- It also provides a basis for application programs and acts as an intermediary between a 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.

Goals of an Operating System

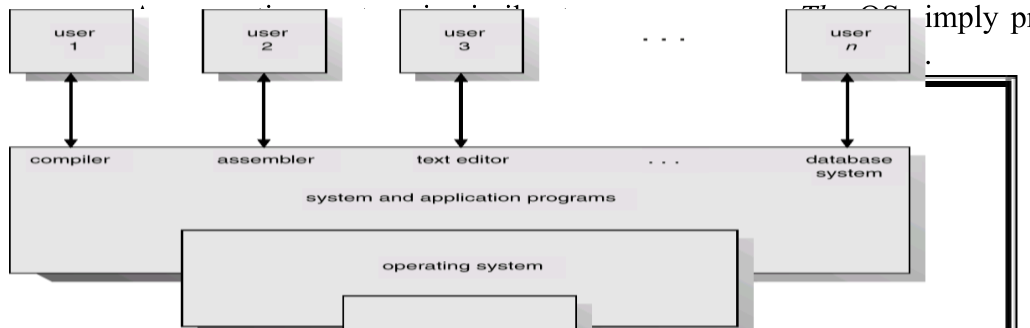
- The primary goal of an operating system is thus to make the computer system convenient to use.
- The secondary goal is to use the computer hardware in an efficient manner.

Components of a Computer System

- An operating system is an important part of almost every computer system.
- A computer system can be divided roughly into four components.
  i. Hardware
  ii. Operating system
  iii. The application programs
  iv. Users

OPERATING SYSTEM STRUCTURE & STRATEGIES

- The hardware - the central processing unit **(CPU),** the memory, and the Input/output **(I/O***)* devices-provides the basic computing resources.
- The application programs- such as word processors, spreadsheets, compilers, and web browsers- define the ways in which these resources are used to solve the computing problems of the users.
- imply provides an .



- The hardware - the central processing unit **(CPU),** the memory, and the Input/output **(I/O*)*** devices-provides the basic computing resources.
- The application programs- such as word processors, spreadsheets, compilers, and web browsers- define the ways in which these resources are used to solve the computing problems of the users.
- An operating system is similar to a *government. The* OS simply provides an environment within which other programs can do useful work.

Abstract view of the components of a computer system.

- Operating system can be viewed as a resource allocator.
- The OS acts as the manager of the resources (such as CPU time, memory space, files storage space, I/O devices) and allocates them to specific programs and users as necessary for tasks.
- An operating system is a control program. It controls the execution of user programs to prevent errors and improper use of computer.

<u>Generations of operating systems:</u>

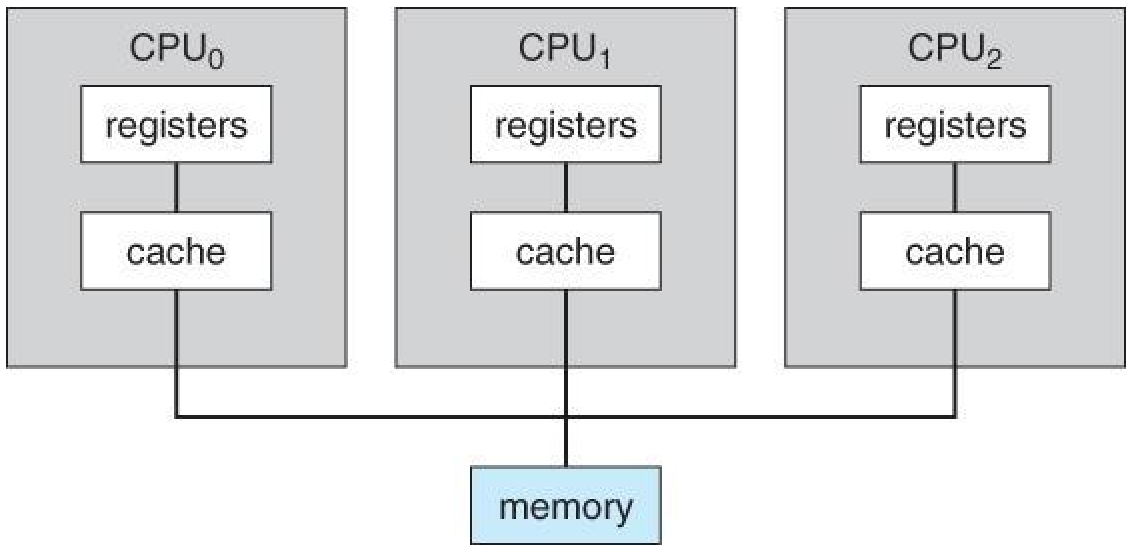
Computer-System Architecture - Different Operating Systems for Different Kinds of Computer Environments

1.3.1 Single-Processor Systems

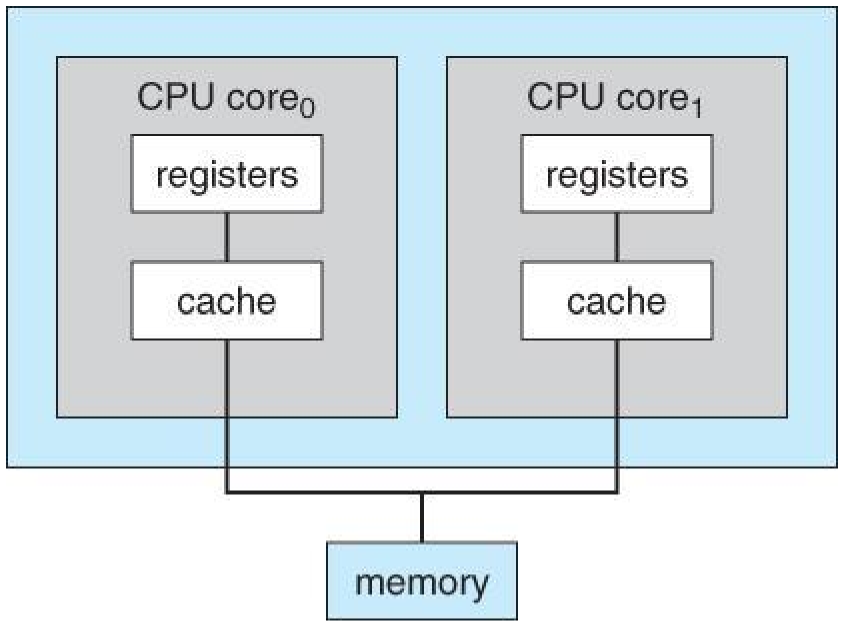
- One main CPU which manages the computer and runs user apps.
- Other specialized processors (disk controllers, GPUs, etc.) do not run user apps.

1.3.2 Multiprocessor Systems

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



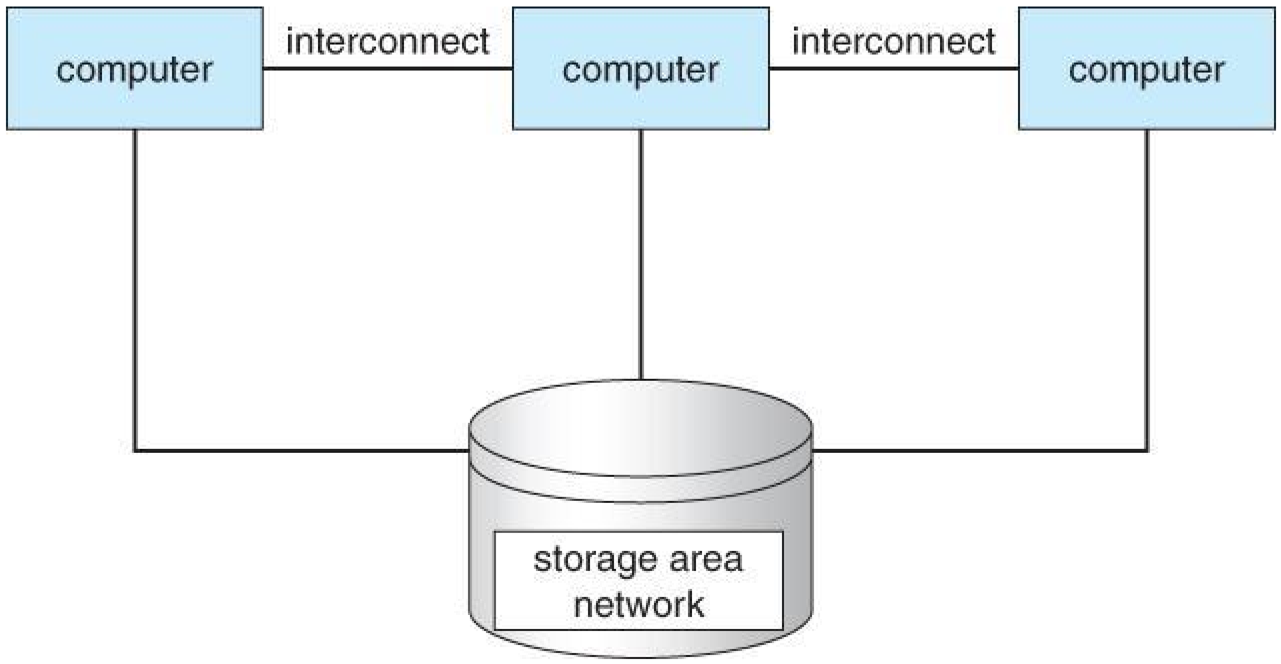
Symmetric multiprocessing architecture



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

Clustered Systems

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



General structure of a clustered system

Generations of operating systems:

1. Serial processing

- No operating system
- Machines run from a console with display lights and toggle switches, input device, and printer
- The programmer interacted directly with the computer hardware.
- Program is read/loaded via input device (eg: card reader)
- Here, the programmers themselves have to schedule the usage of system.

2. Simple Batch Systems

- The user no longer has direct access to the machine. The central idea behind this was the use of a piece of software known as Monitor.

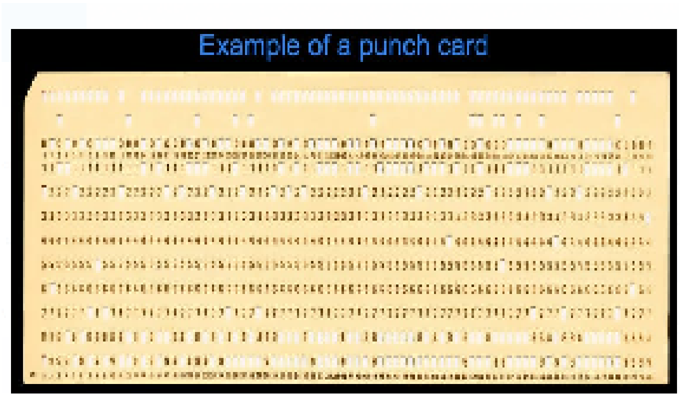
Monitors

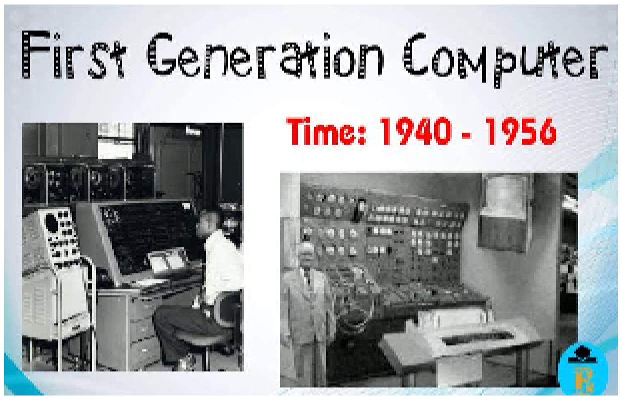
- Software that controls the running programs
- The user submits the job on punch cards or tape to a computer operator, who batches the jobs together.
- The operator places the entire batch on an input device for use by monitor.
- Monitor reads one program
- Program branches back to monitor when finished.
- Then it immediately reads the next job.
- Results of each program are printed on the output device(printer)

- Resident monitor is in main memory and available for execution

Thus monitor performs a scheduling function: A batch of jobs is queued up, and jobs are executed as rapidly as possible, with no intervening idle time. The monitor improves job setup time as well. With each job, instructions are included in a primitive form of job control language (JCL) . This is a special type of programming language used to provide instructions to the monitor. A simple example is that of a user submitting a program written in the programming language FORTRAN plus some data to be used by the program. All FORTRAN instructions and data are on a separate punched card or a separate record on tape.

But when the program is loading or when the output is being sent to printer, the processor is idle. Moreover, input output devices are very slow compared to the processor.





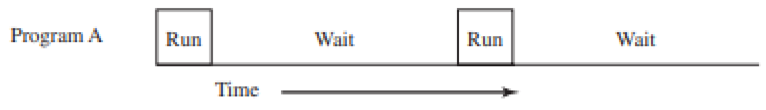
Multiprogrammed Batch Systems Even with the automatic job sequencing provided by a simple batch OS, the processor is often idle. The problem is that I/O devices are slow compared to the processor.

Read one record from file	15 ms
Execute 100 instructions	1 ms
Write one record to file	15 ms
Total	31 ms

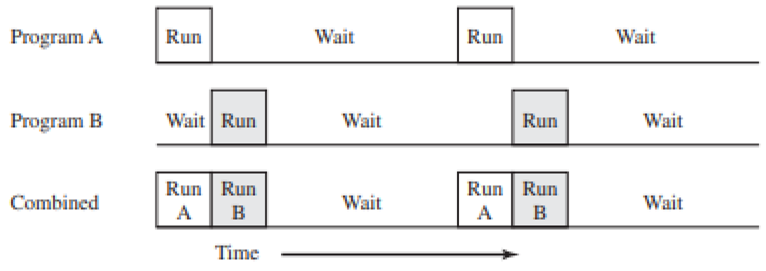
Percent CPU Utilization = 1/ 31 = 0.032 = 3.2%

The calculation concerns a program that processes a file of records and performs, on average, 100 machine instructions per record. In this example, the computer spends over 96% of its time waiting for I/O devices to finish transferring data to and from the file. The Figure below illustrates this situation, where we have a single program, referred to as uniprogramming.

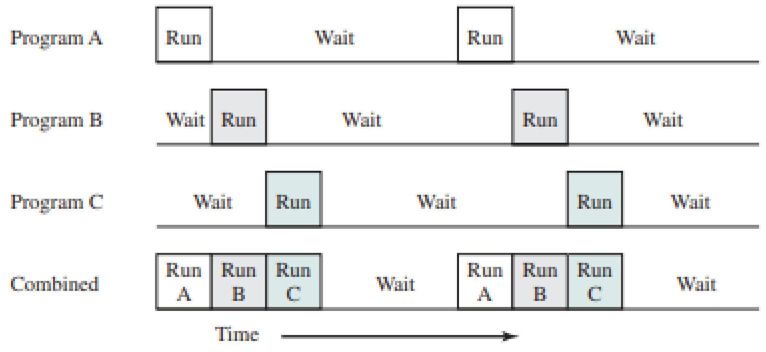
The processor spends a certain amount of time executing, until it reaches an I/O instruction. It must then wait until that I/O instruction concludes before proceeding. This inefficiency is not necessary. We know that there must be enough memory to hold the OS (resident monitor) and one user program. Suppose that there is room for the OS and two user programs. When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O (Figure b). Furthermore, we might expand memory to hold three, four, or more programs and switch among all of them (Figure c). The approach is known as multiprogramming, or multitasking . It is the central theme of modern operating systems.



(a) Uniprogramming



(b) Multiprogramming with two programs



(c) Multiprogramming with three programs

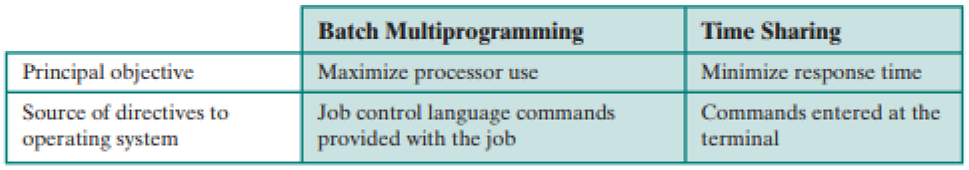
Today, the requirement for an interactive computing facility can be, and often is, met by the use of a dedicated personal computer or workstation. That option was not available in the 1960s, when most computers were big and costly. Instead, time sharing was developed.

Just as multiprogramming allows the processor to handle multiple batch jobs at a time, multiprogramming can also be used to handle multiple interactive jobs. In this latter case, the technique is referred to as **time sharing**, because processor time is shared among multiple users.

In a time-sharing system, multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation. Thus, if there are n users actively requesting service at one time, each user will only see on the average 1/ n of the effective computer capacity, not counting OS overhead. However, given the relatively slow human reaction time, the response time on a properly designed system should be similar to that on a dedicated computer. Both batch processing and time sharing use multiprogramming. The key differences are listed in Table below.

One of the first time-sharing operating systems to be developed was the Compatible Time-Sharing System (CTSS) [CORB62], developed at MIT by a group known as Project MAC (Machine-Aided Cognition, or Multiple-Access Computers). The system was first developed for the IBM 709 in 1961 and later transferred to an IBM 7094.

Batchmultiprogramming versus Time Sharing



	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Distributed Systems

- In contrast to the tightly coupled systems, the processors do not share memory or a clock. Instead , each processor has its own local memory.
- The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. These systems are usually referred to as loosely coupled systems, or distributed systems.

Advantages of distributed systems

- Resource Sharing
- Computation speedup
- Reliability
- Communication

Real-Time Systems

- Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are real-time systems. Some automobile-engine fuel-injection systems, home-appliance controllers, and weapon systems are also real-time systems. A real-time system has well-

defined, fixed time constraints.
- Real-time systems come in two flavours: hard and soft.
- A hard real-time system (also known as an immediate real-time system) is hardware or software that must operate within the confines of a stringent deadline. The application may be considered to have failed if it does not complete its function within the allotted time span.
- A less restrictive type of real-time system is a soft real-time system, where a critical real-time task gets priority over other tasks, and retains that priority until it completes.
- A soft real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirement. In a soft real-time system, the meeting of deadline is not compulsory for every task, but the process should get processed and give the result.
- Soft real-time systems, however, have more limited utility than hard real- time systems. They are useful, in several areas, including multimedia, virtual reality, and advanced scientific projects.

Operating-System Services

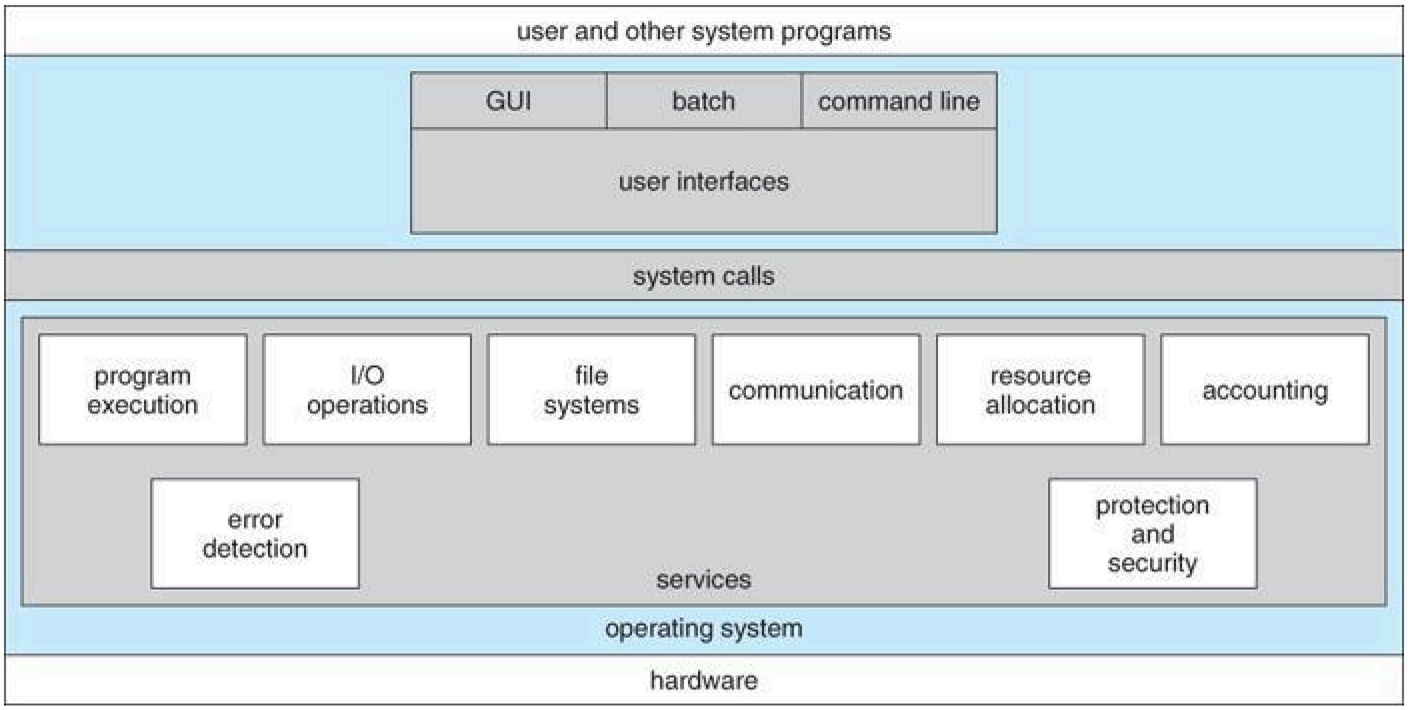


Figure 2.1 - A view of operating system services

OSes provide environments in which programs run, and services for the users of the system, including:

- **User Interfaces** - Means by which users can issue commands to the system. Depending on the system these may be a command-line interface (e.g. sh, csh, ksh, tcsh, etc.), a GUI interface (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a batch command systems. The latter are generally older systems using punch cards of job-control language, JCL, but may still be used today for specialty systems designed for a single purpose.
- **Program Execution** - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.
- **I/O Operations** - The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and storage devices.
- **File-System Manipulation** - In addition to raw data storage, the OS is also responsible for maintaining directory and subdirectory structures, mapping file names to specific blocks of data storage, and providing tools for navigating and utilizing the file system.
- **Communications** - Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented as either shared memory or message passing, (or some systems may offer both.)
- **Error Detection** - Both hardware and software errors must be detected and handled appropriately, with a minimum of harmful repercussions. Some systems may include complex error avoidance or recovery systems, including backups, RAID drives, and other redundant systems. Debugging and diagnostic tools aid users and administrators in tracing down the cause of problems.

Other systems aid in the efficient operation of the OS itself:

- **Resource Allocation** - E.g. CPU cycles, main memory, storage space, and peripheral devices. Some resources are managed with generic systems and others with very carefully designed and specially tuned systems, customized for a particular resource and operating environment.
- **Accounting** - Keeping track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.
- **Protection and Security** - Preventing harm to the system and to resources, either through wayward internal processes or malicious outsiders. Authentication, ownership, and restricted access are obvious parts of this system. Highly secure systems may log all process activity down to excruciating detail, and security regulation dictate the storage of those records on permanent non-erasable medium for extended times in secure (off-site) facilities.

Operating-System Structure

For efficient performance and implementation an OS should be partitioned into separate subsystems, each with carefully defined tasks, inputs, outputs, and performance characteristics. These subsystems can then be arranged in various architectural configurations:

2.7.1 Simple Structure-Monolithic structure

When DOS was originally written its developers had no idea how big and important it would eventually become. It was written by a few programmers in a relatively short amount of time, without the benefit of modern software engineering techniques, and then gradually grew over time to exceed its original expectations. It does not break the system into subsystems, and has no distinction between user and kernel modes, allowing all programs direct access to the underlying hardware. (Note that user versus kernel mode was not supported by the 8088 chip set anyway, so that really wasn't an option back then.)

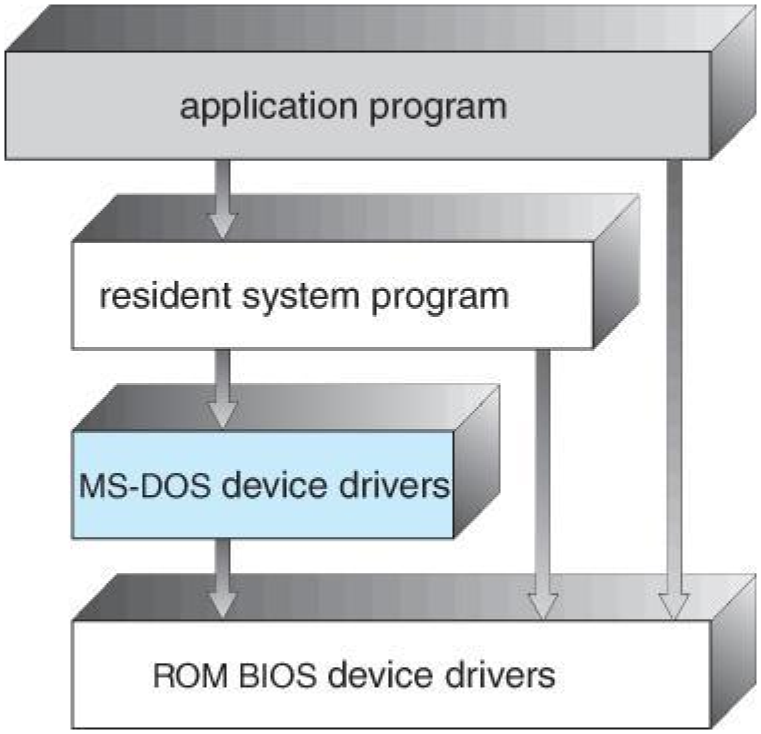


Figure 2.11 - MS-DOS layer structure

The original UNIX OS used a simple layered approach, but almost all the OS was in one big layer, not really breaking the OS down into layered subsystems:

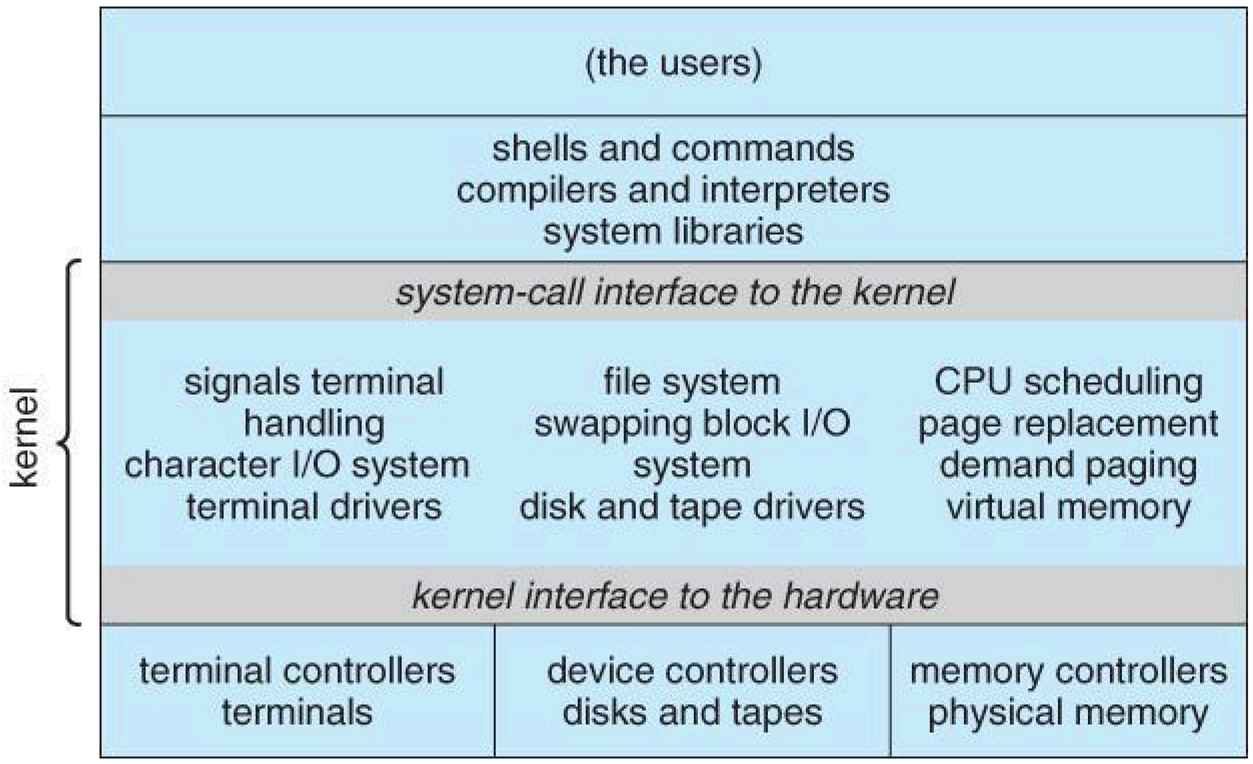


Figure 2.12 - Traditional UNIX system structure

2.7.2 Layered Approach

- Another approach is to break the OS into a number of smaller layers, each of which rests on the layer below it, and relies solely on the services provided by the next lower layer.
- This approach allows each layer to be developed and debugged independently, with the assumption that all lower layers have already been debugged and are trusted to deliver proper services.
- The problem is deciding what order in which to place the layers, as no layer can call upon the services of any higher layer, and so many chicken-and-egg situations may arise.
- Layered approaches can also be less efficient, as a request for service from a higher layer has to filter through all lower layers before it reaches the HW, possibly with significant processing at each step.

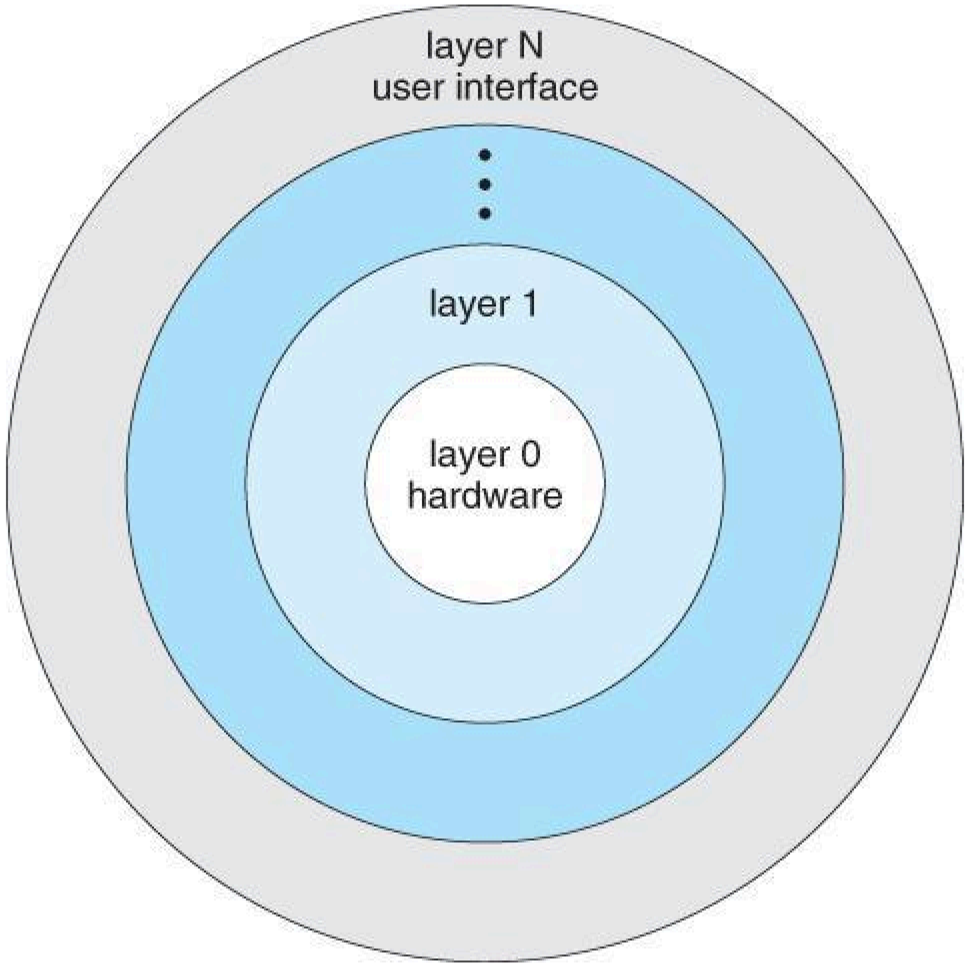


Figure 2.13 - A layered operating system

2.7.3 Microkernels

- The basic idea behind micro kernels is to remove all non-essential services from the kernel, and implement them as system applications instead, thereby making the kernel as small and efficient as possible.
- Most microkernels provide basic process and memory management, and message passing between other services, and not much more.
- Security and protection can be enhanced, as most services are performed in user mode, not kernel mode.
- System expansion can also be easier, because it only involves adding more system applications, not rebuilding a new kernel.
- Mach was the first and most widely known microkernel, and now forms a major component of Mac OSX.
- Windows NT was originally microkernel, but suffered from performance problems relative to Windows 95. NT 4.0 improved performance by moving more services into the kernel, and now XP is back to being more monolithic.
- Another microkernel example is QNX, a real-time OS for embedded systems.

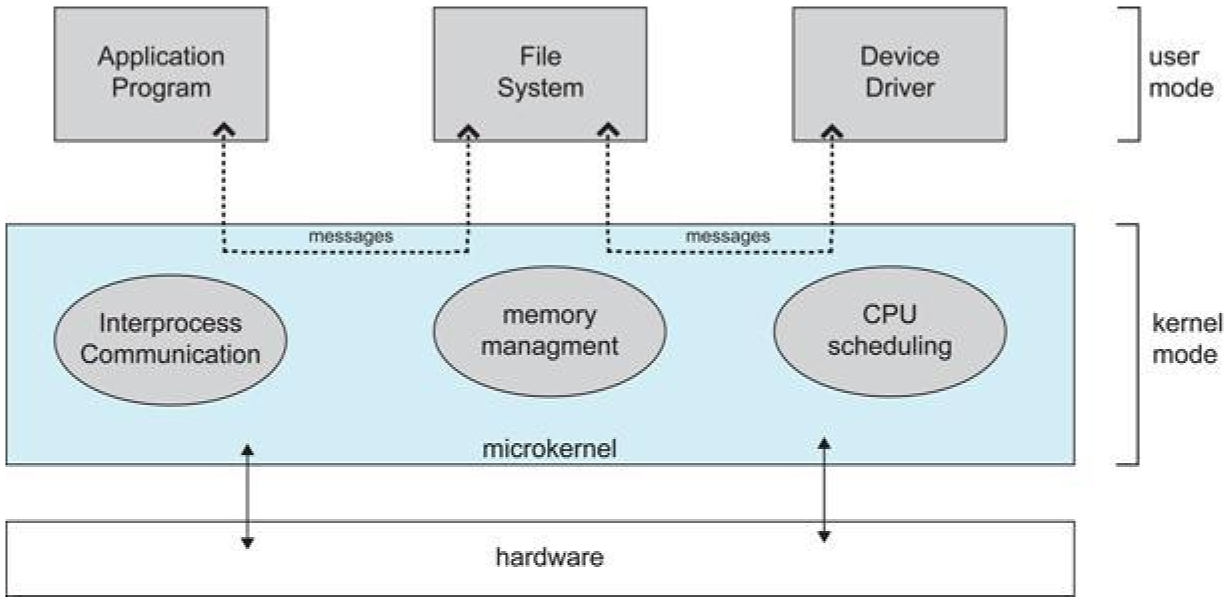


Figure 2.14 - Architecture of a typical microkernel

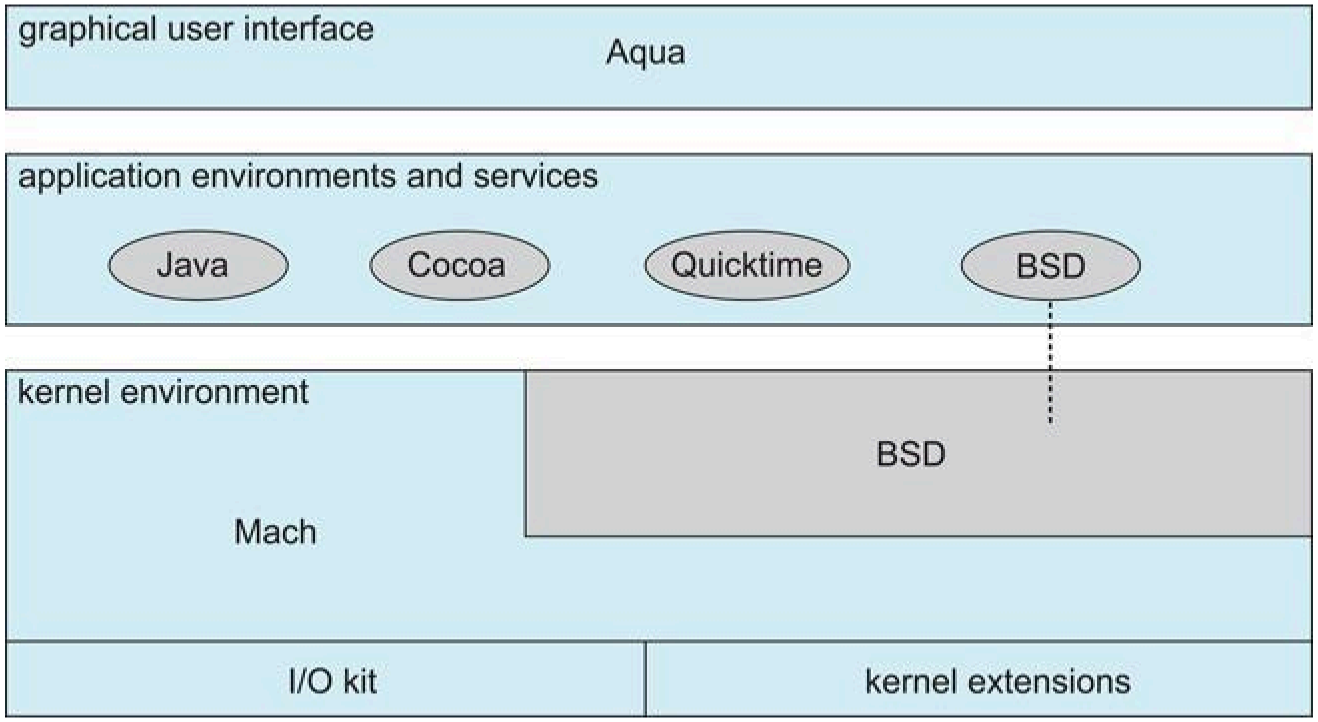


Figure 2.16 - The Mac OS X structure

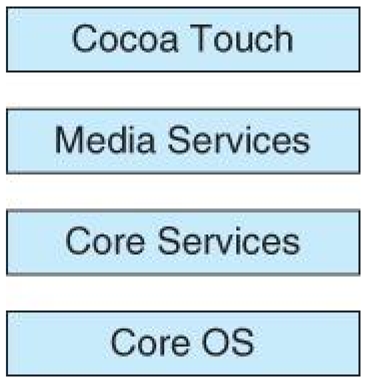


Figure 2.17 - Architecture of Apple's iOS.

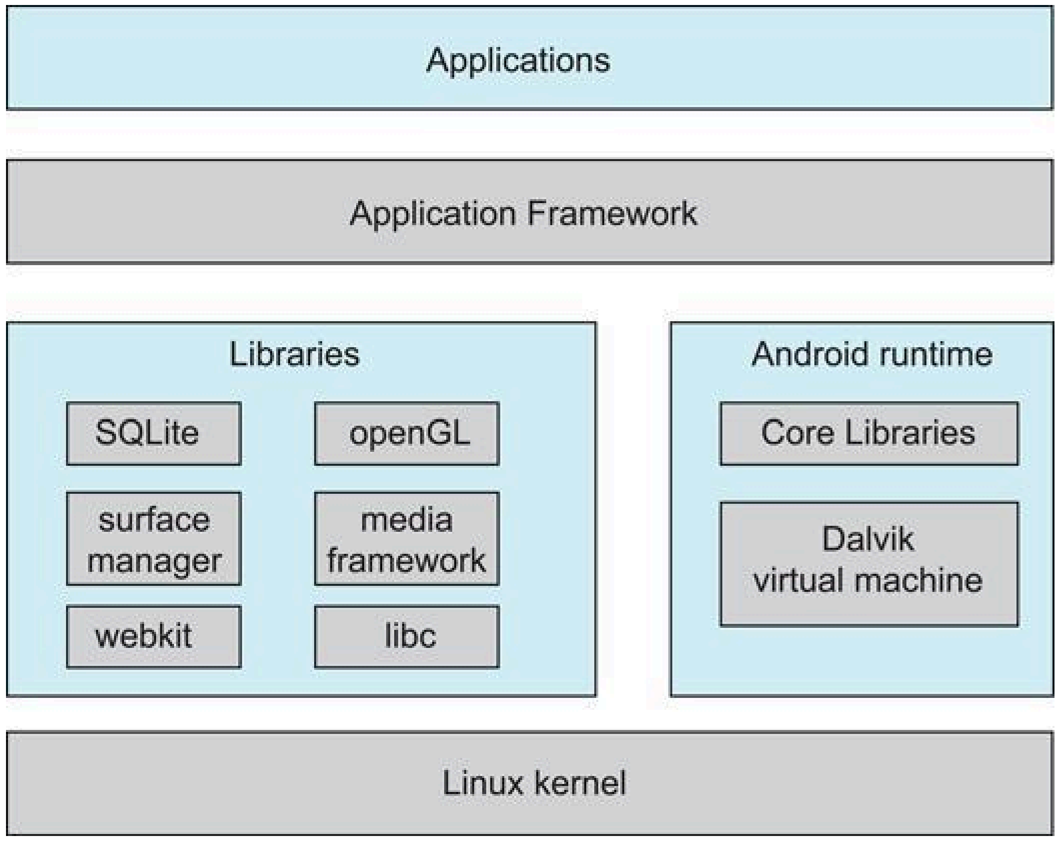


Figure 2.18 - Architecture of Google's Android

System calls

- Kernel: The main component of os that manages operations of computer and hardware.

- User programs will not be able to access any data in the kernel space . then,

How does the user program access it?

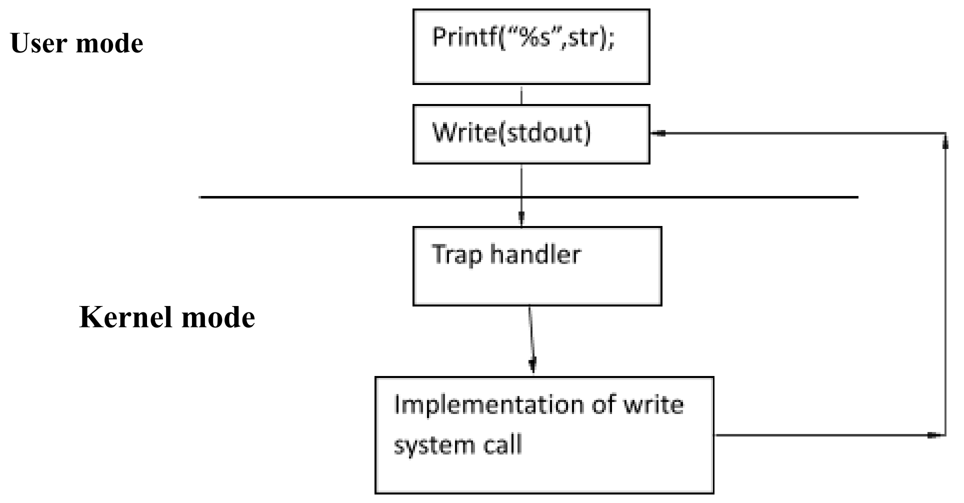
- Using a system call
- System call invokes a function in the kernel using a **Trap**

This causes

- Processor to shift from user mode to privileged mode(kernel mode)
- When it is completed, user mode is resumed.

Kernel mode and user mode:

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. When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the operating system (via a system call), the system must transition from user to kernel mode to fulfill the request.



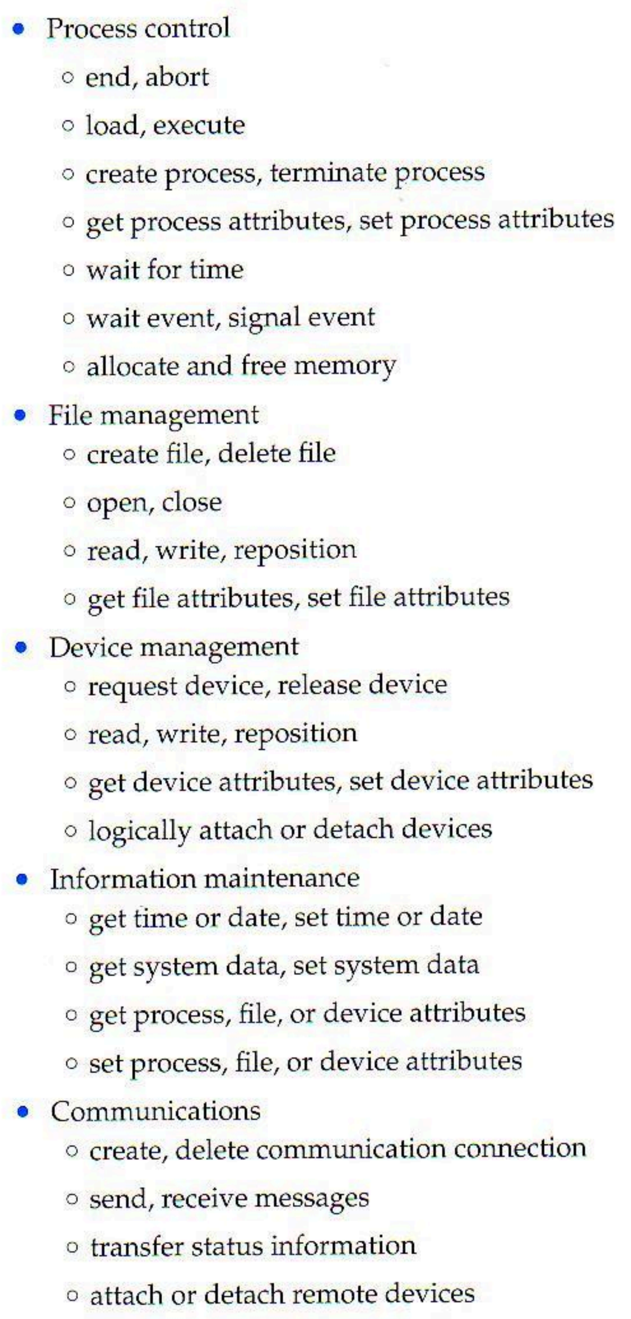
Difference Between System Call And Library Call :

S.no	**SYSTEM CALL**	**PROCEDURE CALL**
1.	A system call is a request made by the program to enter into kernel mode to access a process..	A procedure call is a request made by the program to access a library function defined in a programming library.

2.	In kernel mode the programs are directly accessible to the memory and hardware resources.	In user mode, the programs cannot directly accessible to the memory and hardware resources.
3.	switches from user mode to Kernel mode.	executed in user mode only.
5.	A system call is a function provided by the kernel to enter into the kernel mode to access the hardware resources.	A Library call is a function provided by the programming library to perform a task.
6.	System call are the entry points of the kernel, and therefore they are not linked to the program.	Library functions are linked into your program.
8.	System call have more privileges than library calls because it runs in a supervisory mode.	Library call have less privileges than system calls because it is runs in a user mode only.

Examples of system calls: fork(), exec()

Examples of procedure calls: scanf(), printf();



- Process control
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- 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
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

Figure 2.8 Types of system calls.

Computing Environments

Traditional Computing

Mobile Computing

- Computing on small handheld devices such as smart phones or tablets. (As opposed to laptops, which still fall under traditional computing.)

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

Distributed Systems

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

Client-Server Computing

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

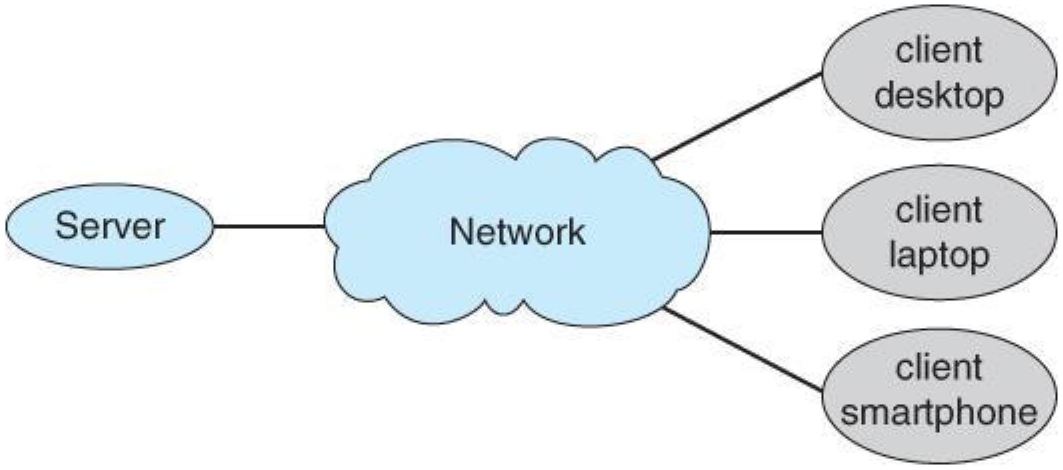


Figure 1.18 - General structure of a client-server system

Peer-to-Peer Computing

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

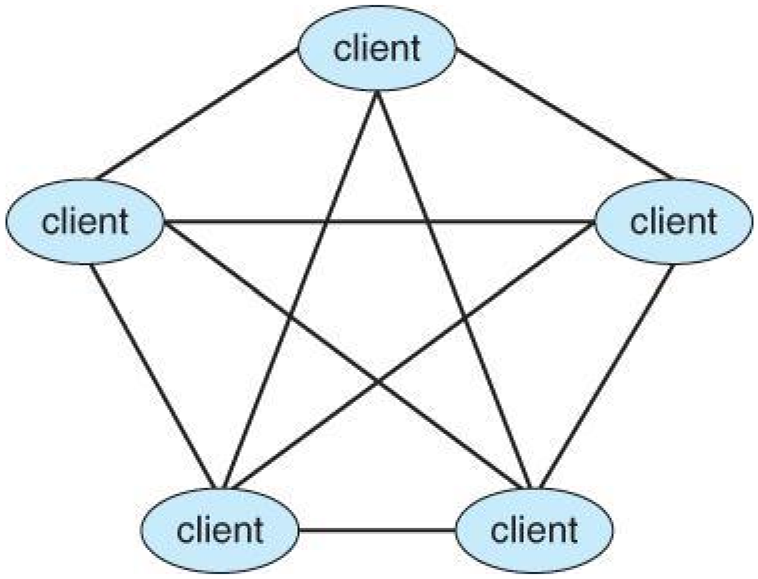


Figure 1.19 - Peer-to-peer system with no centralized service

Virtualization

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

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

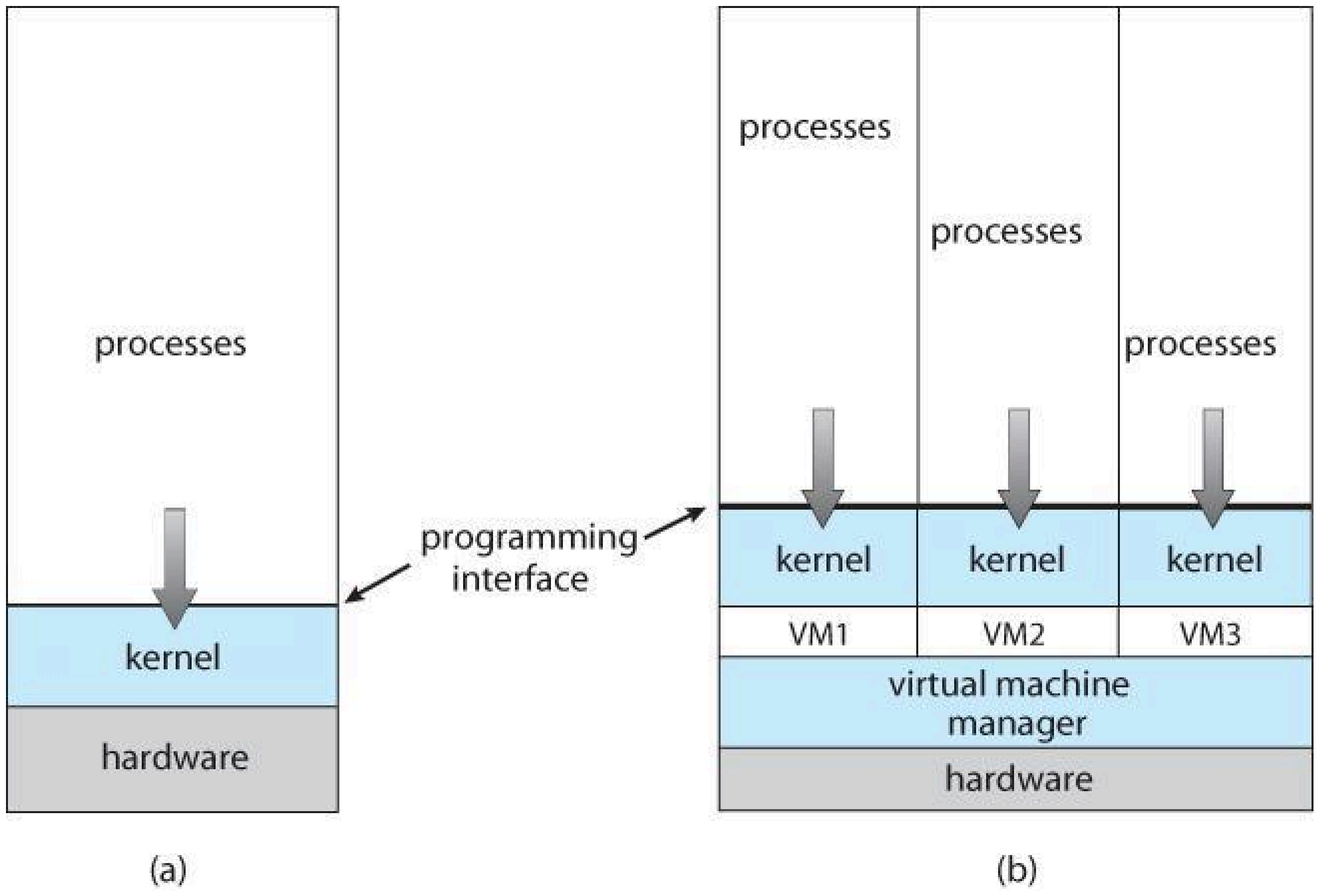


Figure 1.20 - VMWare

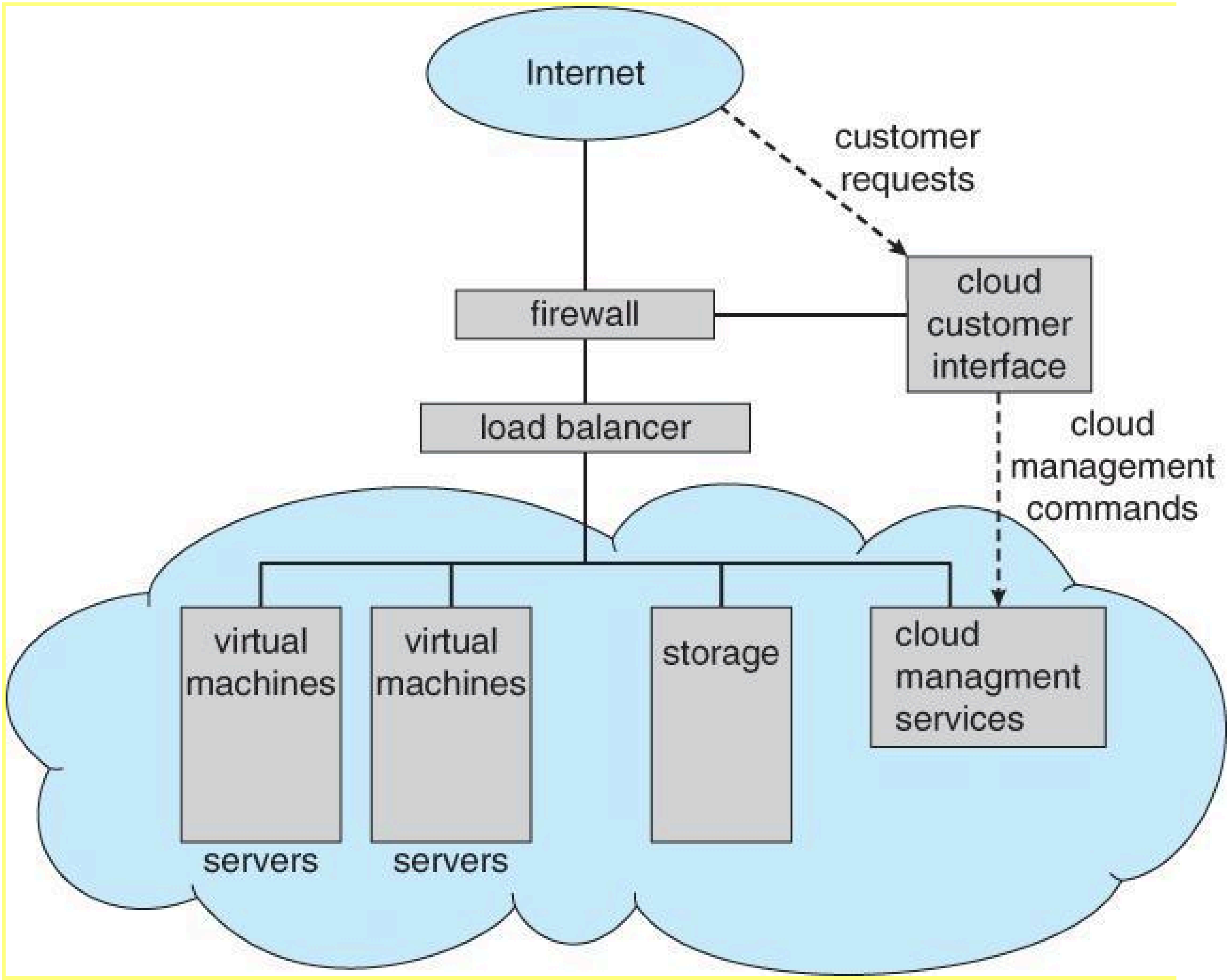
- To run Linux on a Windows system using VMware, follow these steps:
    1. Download the free "VMware Player" tool from http://www.vmware.com/download/player and install it on your system
    2. Choose a Linux version from among hundreds of virtual machine images at http://www.vmware.com/appliances
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

Cloud Computing

- Delivers computing, storage, and applications as a service over a network.

Types of cloud computing:

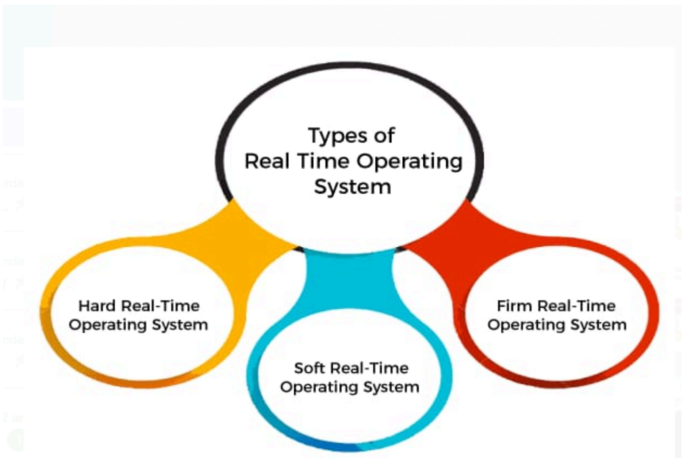
- Public cloud - Available to anyone willing to pay for the service.
- Private cloud - Run by a company for internal use only.
- Hybrid cloud - A cloud with both public and private components.
- Software as a Service - SaaS - Applications such as word processors available via the Internet
- Platform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

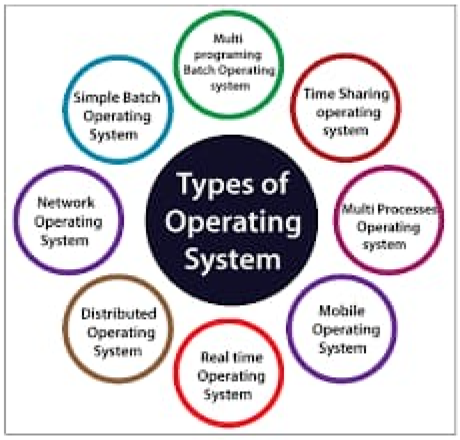


Cloud computing

Real-Time Embedded Systems

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


Types of Real Time Operating System
Hard Real-Time Operating System
Soft Real-Time Operating System
Firm Real-Time Operating System


Types of Operating System
Multi programing Batch Operating system
Time Sharing operating system
Multi Processes Operating system
Mobile Operating System
Real time Operating System
Distributed Operating System
Network Operating System
Simple Batch Operating System