Program : **B.Tech**

Subject Name: **Operating System**

Subject Code: **IT-501**

Semester: **5th**

Chameli Devi Group of Institute
Department of Information Technology

Class Notes

Introduction to operating System

- Operating System is a Resource Manager.
- It is a Control Program.

Definition: Is a collection of software enhancements, executed on the bare hardware, culminating in a high-level virtual machine that serves as an advanced programming environment

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.
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.

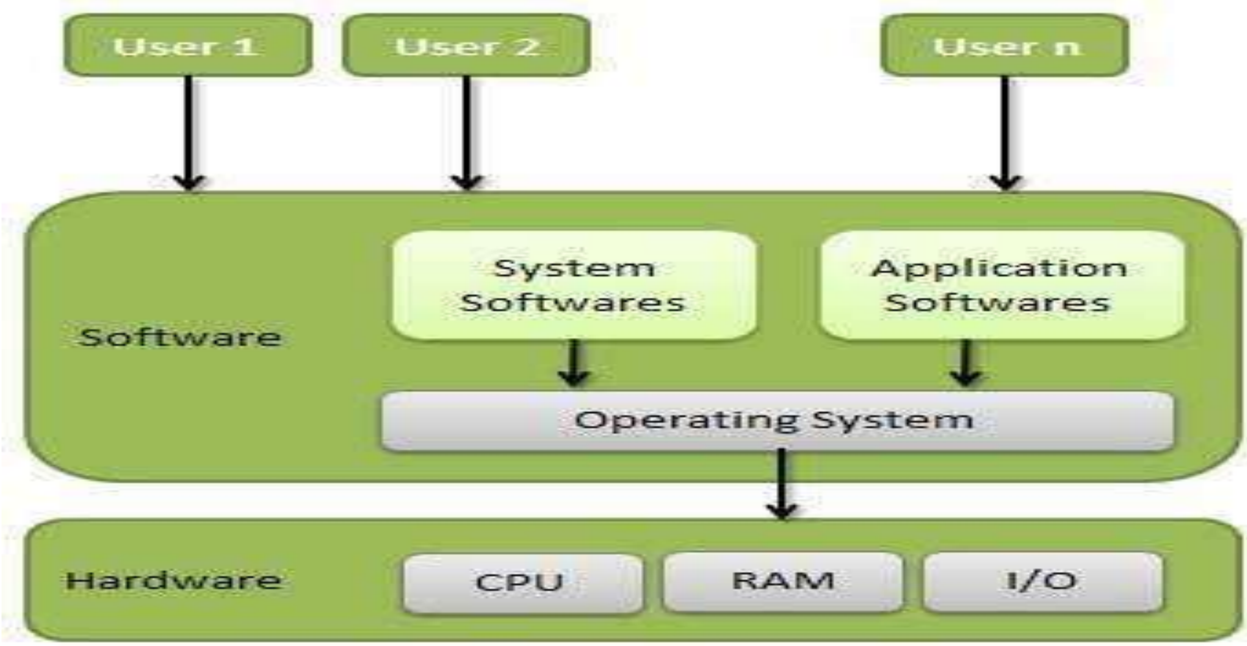

Fig 1.1 OS

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

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be access directly by the CPU. For a program to be executed it must in the main memory. An Operating System does the following activities for memory management −

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what parts are not in use
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so
- De-allocates the memory when a process no longer needs it or has been terminated

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. An Operating System does the following activities for processor management −

- Keeps tracks of processor and status of process the program responsible for this task is known as traffic controller
- Allocates the processor (CPU) to a process
- De-allocates processor when a process is no longer required

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management −

- Keeps tracks of all devices program responsible for this task is known as the I/O controller
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way
- De-allocates devices

File Management

A file system is normally organized into directories for easy navigation and usage these directories may contain files and other directions.

An Operating System does the following activities for file management −

- Keeps track of information, location, uses, status etc the collective facilities are often known as file system
- Decides who gets the resources
- Allocates the resources
- De-allocates the resources

Other Important Activities

Following are some of the important activities that an Operating System performs −

- **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** − Recording delays between request for a service and response from the system.
- **Job accounting** − Keeping track of time and resources used by various jobs and users
- **Error detecting aids** − Production of dumps, traces, error messages, and other debugging and error detecting aids
- **Coordination between other software and users** − Coordination and assignment of compilers interpreters, assemblers and other software to the various users of the computer systems

Need of Operating System

- Computer hardware is developed to execute user programs and make solving user problems easier
- An operating system makes a computer more convenient to use.

It acts as an interface between user and computer hardware. Therefore, the end-users are not particularly concerned with the computer's architecture, and they view the computer system in terms of an application.

To programmers, it provides some basic utilities to assist him in creating programs, the management of files, and the control of I/O devices.

Operating System Objectives

- Convenience
- Efficiency
- Ability to evolve

Services Provided by Operating Systems

- Facilities for program creation
- Program execution
- Access to I/O and files
- System access
- Error detection and response
    - Internal and external hardware errors
        - Memory error
        - Device failure
    - Software errors
        - Arithmetic overflow
        - Access forbidden memory locations
    - Operating system cannot grant request of application
- Accounting
    - Collect statistics
    - Monitor performance
    - Used to anticipate future enhancements
    - Used for billing users

Computer System Components

A computer system can be dividing into four components

The Hardware: Provides basic computing resources (CPU, memory, I/O devices)

The Operating System: Controls and coordinates the use of the hardware among the various application programs for the various users.

The Application Programs: Define the ways in which the system resources are use to solve the computing problems of the users (compilers, database systems, video games, business programs)

The Users: Users (people, machines, other computers).These components can be view as a layer where each layer uses the services provided by the layer beneath it

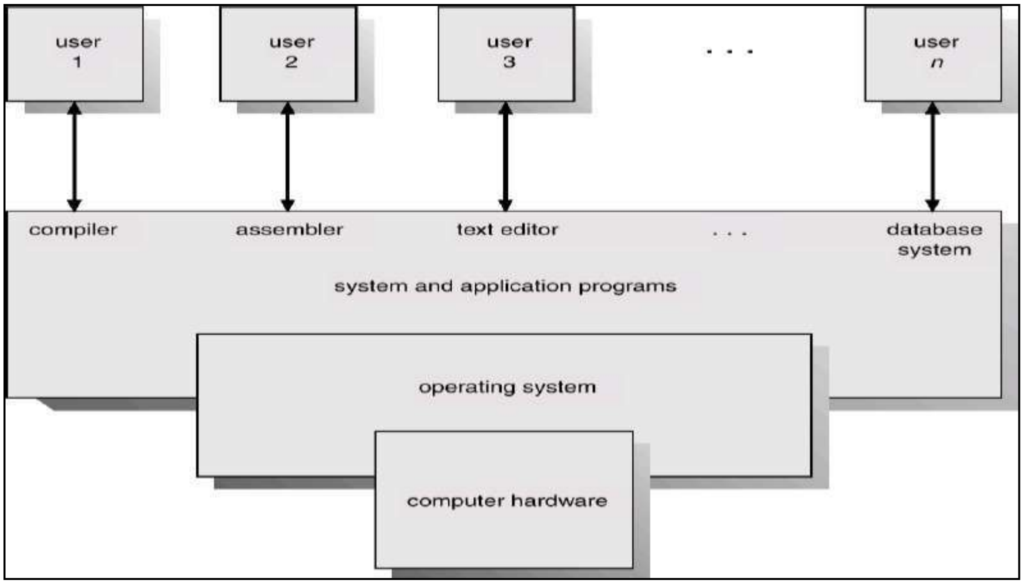

Fig 1.2 Components of OS

Evolution of operating systems

Serial Processing

- Users access the computer in series. From the late 1940's to mid-1950's, the programmer interacted directly with computer hardware i.e., no operating system. These machines were run with a console consisting of display lights, toggle switches, some form of input device and a printer.
- Programs in machine code are loaded with the input device like card reader. If an error occur the program was halted, and the error condition was indicated by lights. Programmers examine the registers and main memory to determine error. If the program is success, then output will appear on the printer.
- Main problem here is the setup time. That is single program needs to load source program into memory, saving the compiled (object) program and then loading and linking together.

Simple Batch Systems

To speed up processing, jobs with similar needs are batched together and run as a group. Thus, the programmers will leave their programs with the operator. The operator will sort programs into batches with similar requirements.

The problems with Batch Systems are:

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.

For overcoming this problem use the Spooling Technique. Spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. That is when the job requests the printer to output a line, that line is copied into a system buffer and is written to the disk. When the job is completed, the output is printed. Spooling technique can keep both the CPU and the I/O devices working at much higher rates.

Multiprogrammed Batch Systems

- Jobs must be run sequentially, on a first come, first-served basis. However, when several jobs are on a direct-access device like disk, job scheduling is possible. The main aspect of job scheduling is

multiprogramming. Single user cannot always keep the CPU or I/O devices busy. Thus, multiprogramming increases CPU utilization.
- In when one job needs to wait, the CPU is switched to another job, and so on. Eventually, the first job finishes waiting and gets the CPU back.

Time-Sharing Systems

- Time-sharing systems are not available in 1960s. Time-sharing or multitasking is a logical extension of multiprogramming. That is processors time is shared among multiple users simultaneously is called time-sharing. The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is in Multiprogrammed batch systems its objective is maximize processor use, whereas in Time-Sharing Systems its objective is minimize response time.
- Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is seconds at most.
- Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Personal-Computer Systems (PCs)

- A computer system is dedicated to a single user is called personal computer, appeared in the 1970s. Micro computers are considerably smaller and less expensive than mainframe computers. The goals of the operating system have changed with time; instead of maximizing CPU and peripheral utilization, the systems developed for maximizing user convenience and responsiveness. For e.g., MS-DOS, Microsoft Windows and Apple Macintosh.
- Hardware costs for microcomputers are sufficiently low. Decrease the cost of computer hardware (such as processors and other devices) will increase our needs to understand the concepts of operating system. Malicious programs destroy data on systems. These programs may be self-replicating and may spread rapidly via worm or virus mechanisms to disrupt entire companies or even worldwide networks.
- MULTICS operating system was developed from 1965 to 1970 at the Massachusetts Institute of Technology (MIT) as a computing utility. Many of the ideas in MULTICS were subsequently used at Bell Laboratories in the design of UNIX OS.

Parallel Systems

Most systems to date are single-processor systems; that is they have only one main CPU. Multiprocessor systems have more than one processor.

The advantages of parallel system are as follows:

- throughput (Number of jobs to finish in a time period)
- Save money by sharing peripherals, cabinets and power supplies
- Increase reliability
- Fault-tolerant (Failure of one processor will not halt the system).

Symmetric multiprocessing model

- Each processor runs an identical job (copy) of the operating system, and these copies communicate. Encore's version of UNIX operating system is a symmetric model.
- E.g., If two processors are connected by a bus. One is primary and the other is the backup. At fixed check points in the execution of the system, the state information of each job is copied from the primary machine to the backup. If a failure is detected, the backup copy is activated, and is restarted from the most recent checkpoint. But it is expensive.

Asymmetric multiprocessing model

Each processor is assigned a specific task. A master processor controls the system. Sun's operating system SunOS version 4 is an asymmetric model. Personal computers contain a microprocessor in the keyboard to convert the keystrokes into codes to be sent to the CPU.

Distributed Systems

Distributed systems distribute computation among several processors. In contrast to tightly coupled systems (i.e., parallel 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 are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers and so on.

The advantages of distributed systems are as follows:

- Resource Sharing: With resource sharing facility user at one site may be able to use the resources available at another.
- Communication Speedup: Speedup the exchange of data with one another via electronic mail.
- Reliability: If one site fails in a distributed system, the remaining sites can potentially continue operating.

Real-time Systems

Real-time systems are used when there are rigid time requirements on the operation of a processor, or the flow of data and real-time systems can be used as a control device in a dedicated application. Real-time operating system has well-defined, fixed time constraints otherwise system will fail.

E.g., Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers.

There are two types of real-time systems:

- **Hard real-time systems**
  Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found.

- **Soft real-time systems**
  Soft real time systems are less restrictive. Critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems.

  E.g., Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers.

Types of OS

Operating systems are there from the very first computer generation and they keep evolving with time. In this chapter, we will discuss some of the important types of operating systems which are most commonly used.

Batch operating system

The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows −

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

Time-sharing is a technique which enables many people, located at various terminals, to use a computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if **n** users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time, Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are as follows −

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows −

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows −

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows −

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the **response time**. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, scientific experiments, medical image systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing, and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects likes undersea exploration and planetary rovers, etc.

System Protection:

Goals of Protection

- To prevent malicious misuse of the system by users or programs.
- To ensure that each shared resource is used only in accordance with system *policies,* which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the *mechanisms* for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

Principles of Protection

- The ***principle of least privilege*** dictates that programs, users, and systems be given just enough privileges to perform their tasks.
- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of "network" or "backup" or some other pseudo group, rather than SUID with root ownership. This limits the amount of damage that can occur if something goes wrong.
- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities - The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges

Operating system structure:

An operating system might have many structures. According to the structure of the operating system; operating systems can be classified into many categories.

Some of the main structures used in operating systems are:

1. **Monolithic architecture of operating system**

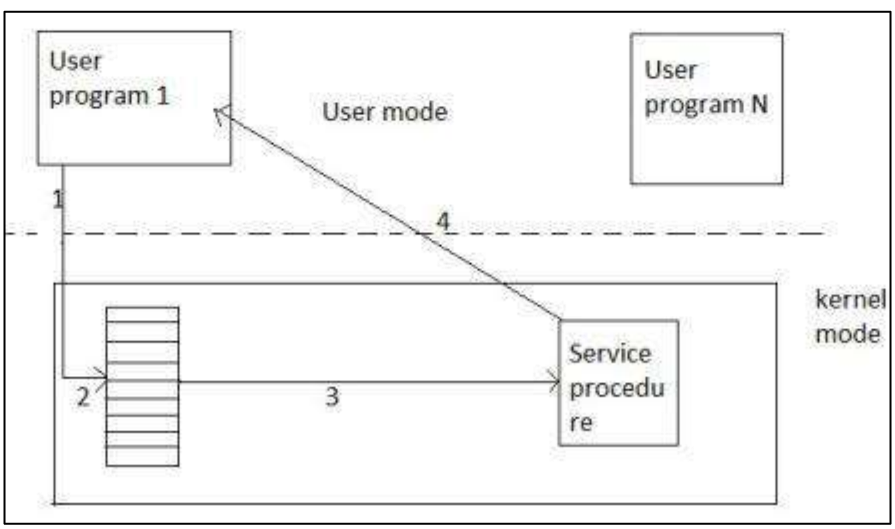

Fig 1.3: monolithic structure of operating system

It is the oldest architecture used for developing operating system. Operating system resides on kernel for anyone to execute. System call is involved i.e. Switching from user mode to kernel mode and transfer control to operating system shown as event 1. Many CPU has two modes, kernel mode, for the operating system in which all instruction are allowed and user mode for user program in which I/O devices and certain other instruction are not allowed. Two operating system then examines the parameter of the call to determine which system call is to be carried out shown in event 2. Next, the operating system index's into a table that contains procedure that carries out system call. This operation is shown in events. Finally, it is called when the work has been completed and the system call is finished, control is given back to the user mode as shown in event 4.

2. **Layered Architecture of operating system**

The layered Architecture of operating system was developed in 60's in this approach; the operating system is broken up into number of layers. The bottom layer (layer 0) is the hardware layer and the highest layer (layer n) is the user interface layer as shown in the figure.

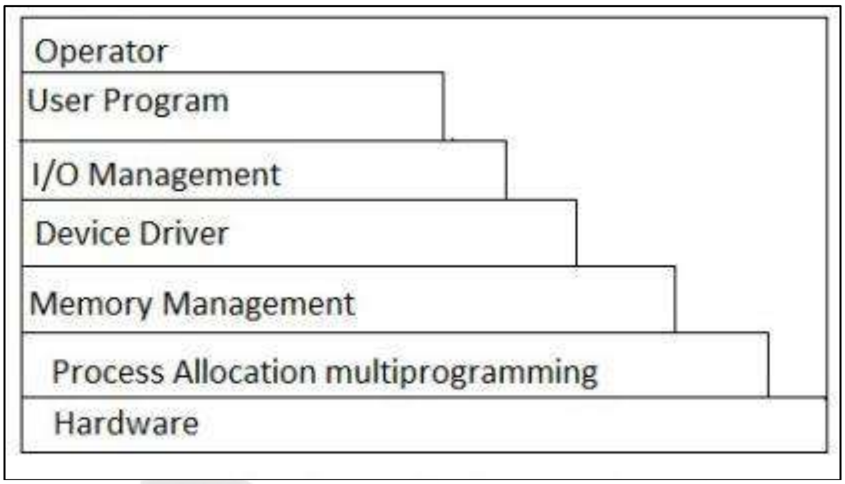

Fig 1.4: layered architecture

- The layered are selected such that each user functions and services of only lower level layer. The first layer can be debugged wit out any concern for the rest of the system. Its user basic hardware to implement this function once the first layer is debugged., it's correct functioning can be assumed while the second layer is debugged & soon. If an error is found during the debugged of particular layer, the layer must be on that layer, because the layer below it already debugged. Because of this design of the system is simplified when operating system is broken up into layer.
- Os/2 operating system is example of layered architecture of operating system another example is earlier version of Windows NT.
- The main disadvantage of this architecture is that it requires an appropriate definition of the various layers & a careful planning of the proper placement of the layer.

3. **Virtual memory architecture of operating system**

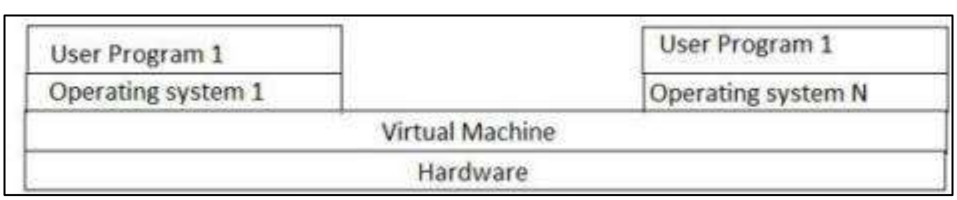

Fig 1.5: virtual memory architecture

Virtual machine is an illusion of a real machine. It is created by a real machine operating system, which make a single real machine appears to be several real machines. The architecture of virtual machine is shown above. The best example of virtual machine architecture is IBM 370 computer. In this system each user can choose a different operating system. Virtual machine can run several operating systems at once, each of them on its virtual machine.

Its multiprogramming shares the resource of a single machine in different manner.
The concepts of virtual machine are: -

a. Control program (cp): - cp creates the environment in which virtual machine can execute. It gives to each user facilities of real machine such as processor, storage I/0 devices.
b. conversation monitor system (cons): - cons is a system application having features of developing program. It contains editor, language translator, and various application packages.
c. Remote spooling communication system (RSCS): - provide virtual machine with the ability to transmit and receive file in distributed system.
d. IPCS (interactive problem control system):- it is used to fix the virtual machine software problems.

4. **client/server architecture of operating system**

A trend in modern operating system is to move maximum code into the higher level and remove as much as possible from operating system, minimizing the work of the kernel. The basic approach is to implement most of the operating system functions in user processes to request a service, such as request to read a particular file, user send a request to the server process, server checks the parameter and finds whether it is valid or not, after that server does the work and send back the answer to client server model works on request- response technique i.e. Client always send request to the side in order to perform the task, and on the other side, server gates complementing that request send back response. The figure below shows client server architecture.

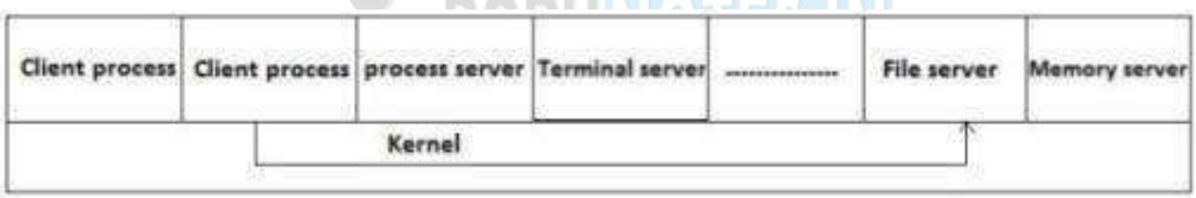

Fig 1.6: client server model

In this model, the main task of the kernel is to handle all the communication between the client and the server by splitting the operating system into number of ports, each of which only handle some specific task. I.e. file server, process server, terminal server and memory service.

Kernel

A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory etc). The main tasks of the kernel are:

Process management

- Device management
- Memory management
- Interrupt handling
- I/O communication
- File system

Types of Kernels

Kernels may be classified mainly in two categories

1. Monolithic
2. Micro Kernel

Monolithic Kernels

Earlier in this type of kernel architecture, all the basic system services like process and memory management, interrupt handling etc. were packaged into a single module in kernel space.

This type of architecture led to some serious drawbacks like

1) Size of kernel, which was huge.

2)Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours in a modern-day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. Linux follows the monolithic modular approach

Microkernels

This architecture majorly caters to the problem of ever-growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file system etc. to run in user space. This reduces the kernel code size and increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional.

System Calls Basics

To understand system calls, first one needs to understand the difference between **kernel mode** and **user mode** of a CPU. Every modern operating system supports these two modes.

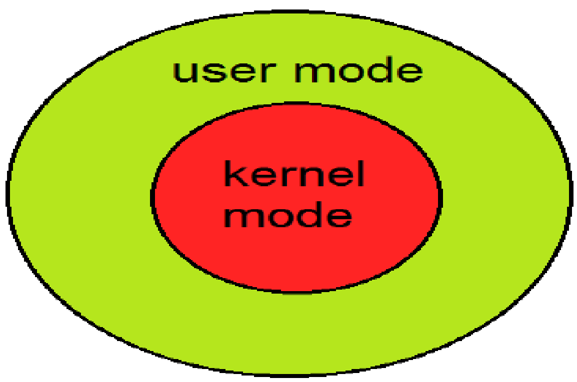

Fig 1.7 Modes supported by the operating system

Kernel Mode

- When CPU is in **kernel mode**, the code being executed can access any memory address and any hardware resource.
- Hence kernel mode is a very privileged and powerful mode.
- If a program crashes in kernel mode, the entire system will be halted.

User Mode

- When CPU is in **user mode**, the programs don't have direct access to memory and hardware resources.
- In user mode, if any program crashes, only that particular program is halted.
- That means the system will be in a safe state even if a program in user mode crashes.
- Hence, most programs in an OS run in user mode.

System Call

When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a **system call**.

When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a **context switch**.

Then the kernel provides the resource which the program requested. After that, another context switching happens which results in change of mode from kernel mode back to user mode.
Generally, system calls are made by the user level programs in the following situations:

- Creating, opening, closing and deleting files in the file system.
- Creating and managing new processes.
- Creating a connection in the network, sending and receiving packets.
- Requesting access to a hardware device, like a mouse or a printer.

System Boot

- Booting the system is done by loading the kernel into main memory and starting its execution.
- The CPU is given a reset event, and the instruction register is loaded with a predefined memory location, where execution starts.
    - The initial bootstrap program is found in the BIOS read-only memory.
    - This program can run diagnostics, initialize all components of the system, loads and starts the Operating System loader. (Called **boot strapping**)
    - The loader program loads and starts the operating system.
    - When the Operating system starts, it sets up needed data structures in memory, sets several registers in the CPU, and then creates and starts the first user level program. From this point, the operating system only runs in response to interrupts.

The following diagram demonstrates the steps involved in a system boot process:

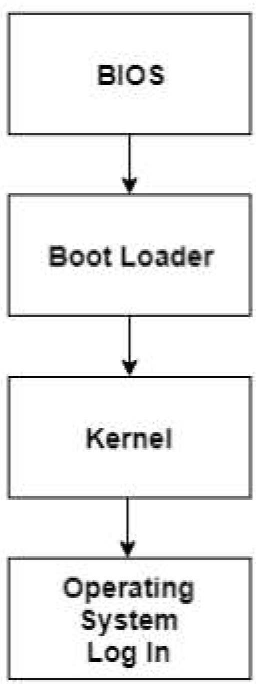

Fig 1.8: steps involved in a system boot process

Operating System Design:

Design Goals

The design changes depending on the type of the operating system i.e. if it is batch system, time shared system, single user system, multiuser system, distributed system etc.

There are basically two types of goals while designing an operating system. These are:

User Goals

The operating system should be convenient, easy to use, reliable, safe and fast according to the users. However, these specifications are not very useful as there is no set method to achieve these goals.

System Goals

The operating system should be easy to design, implement and maintain. These are specifications required by those who create, maintain and operate the operating system. But there is not specific method to achieve these goals as well.

Operating System Implementation

The operating system needs to be implemented after it is designed. Earlier they were written in assembly language, but now higher-level languages are used. The first system not written in assembly language was the Master Control Program (MCP) for Burroughs Computers.

Advantages of Higher-Level Language

There are multiple advantages to implementing an operating system using a higher-level language such as: the code is written faster; it is compact and easier to debug and understand. Also, the operating system can be easily moved from one hardware to another if it is written in a high-level language.

Disadvantages of Higher-Level Language

Using high level language for implementing an operating system leads to a loss in speed and increase in storage requirements. However, in modern systems only a small amount of code is needed for high performance, such as the CPU scheduler and memory manager. Also, the bottleneck routines in the system can be replaced by assembly language equivalents if required.

Spooling –

Spooling stands for Simultaneous peripheral operation online. A spool is a like buffer as it holds the jobs for a device till the device is ready to accept the job. It considers disk as a huge buffer which can store as many jobs for the device till the output devices are ready to accept them.

Buffering –

Main memory has an area called buffer that is used to store or hold the data temporarily that is being transmitted either between two devices or between a device or an application. Buffering is an act of storing data temporarily in the buffer. It helps in matching the speed of the data stream between the sender and receiver. If speed of the sender's transmission is slower than receiver, then a buffer is created in main memory of the receiver, and it accumulates the bytes received from the sender and vice versa.

The basic difference between Spooling and Buffering is that Spooling overlaps the input/output of one job with the execution of another job while the buffering overlaps input/output of one job with the execution of the same job.

Differences between Spooling and Buffering –

- The key difference between spooling and buffering is that Spooling can handle the input/output of one job along with the computation of another job at the same time while buffering handles input/output of one job along with its computation.
- Spooling is a stand for Simultaneous Peripheral Operation online. Whereas buffering is not an acronym.
- Spooling is more efficient than buffering, as spooling can overlap processing two jobs at a time.
- Buffering use limited area in main memory while Spooling uses the disk as a huge buffer.

