OPERATING SYSTEM OVERVIEW

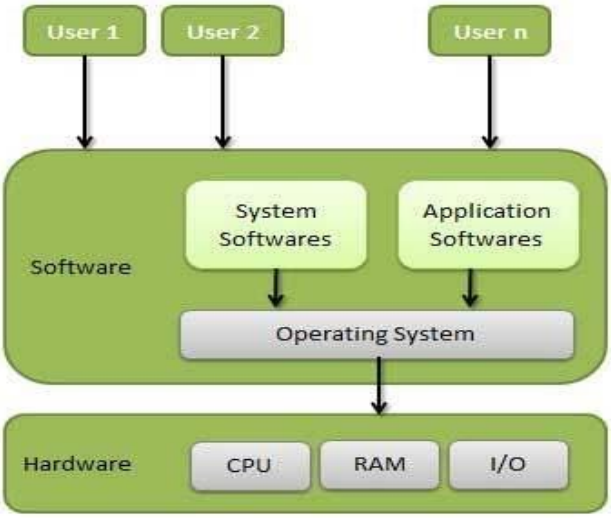
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Another definition:

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Architecture

We can draw a generic architecture diagram of an Operating System which is as follows:



Computer System

• Computer system consists of hardware device and software that are combined to provide a tool to user for solving problems.

• Fig. 1.1.1 shows modern computer system.

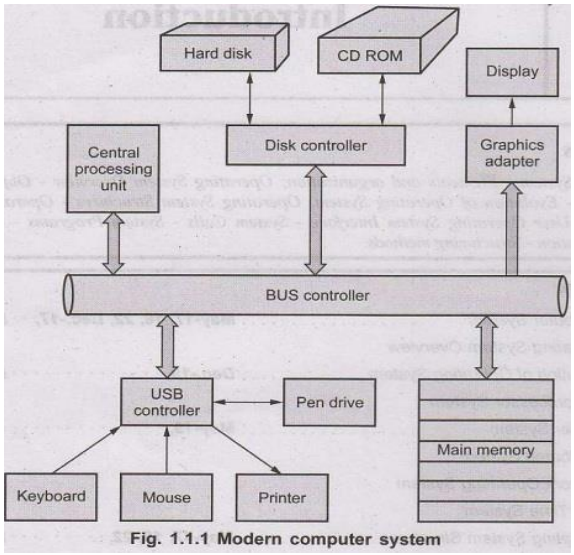


Fig. 1.1.1 Modern computer system

• Modern computer consists of one or two CPU with main memory and various I/O devices. Common bus is used for communication between these devices. Each device has its own device controller.

• CPU and device controller uses memory cycle for execution purposes. But memory cycle is only available to one device at a time.

• Bootstrap program is loaded when user start the computer. It initialies all the device connected to the computer system and then loads required device drivers.

• After this, operating system loads in the computer system. In UNIX OS, an 'init' is the first process which execute by OS.

• Interrupt is software and hardware. It is used to send signal to CPU. Software interrupt is sometime called system call.

• When interrupt is trigger, the CPU stops executing the instruction and control is transfer to the fixed location. Starting address is stored at fixed location where the service routine executes.

• Interrupts do not alter the control flow of the process executing on the processor.

Storage structure

• Processor access the data from main memory before executing any instruction. Main memory is also called Random Access Memory (RAM).

• DRAM is used in main memory. Fig. 1.1.2 shows hierarchy of storage device.

• At the top of the hierarchy, we have storage on the CPU registers. For accessing the CPU, it is fastest form of storage.

• Cache memory capacity is less than 1 MB.

• User program and data are stored in the main memory. Main memory is volatile, so it can not stored permanently.

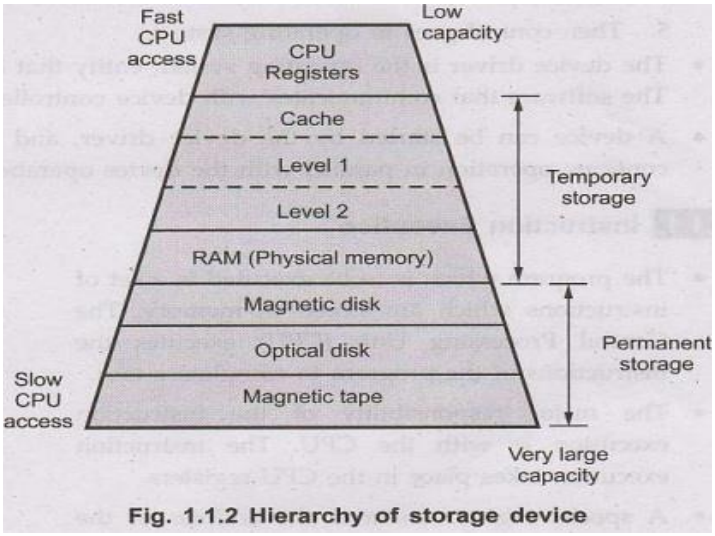


Fig. 1.1.2 Hierarchy of storage device

• Storage system is classified as temporary storage or permanent storage.

• Top level storage devices are low capacity with faster CPU access and bottom level storage devices having very large capacity with slow CPU access speed.

I/O structure

• Every device uses a device controller to connect it to the computer's address and data bus. Devices can be classified as a block oriented or character oriented, depending on the number of bytes transferred on an individual operation.

• Storage devices are used to store data while the computer is off.

• All the I/O devices are connected to each other by using common bus. CPU and main memory is also connected with this bus.

• Various types of controller is used in the computer system. Small Computer System Interface (SCSI) controller can handle upto seven devices. Each device controller have its own buffer.

• Device controller manage the data transfer between peripheral device and its controller. Device driver is handled by device controller.

I/O operation steps

1. Device driver loads the registers within the device controller.

2. Device controller takes action according to the data loaded into the register.

3. Data is transfer from device to its local buffer with the help of device controller.

4. After completion of data transfer, the device controller sends an interrupt signal to device driver about data transfer completion operation.

5. Then control goes to operating system.

• The device driver is the operating system entity that controls CPU-I/O parallelism. The software that communicates with device controller is called device driver.

• A device can be started by the device driver, and the application program can continue operation in parallel with the device operation.

Instruction Execution

• The program which is to be executed is a set of instructions which are stored in memory. The Central Processing Unit (CPU) executes the instructions of the program to complete a task.

• The major responsibility of the instruction execution is with the CPU. The instruction execution takes place in the CPU registers.

• A special register contains the address of the instruction. The CPU "fetches" the instruction from memory at that address.

• The CPU "decodes" the instruction to figure out what to do. The CPU "fetches" any data b(operands) needed by the instruction, from memory or registers.

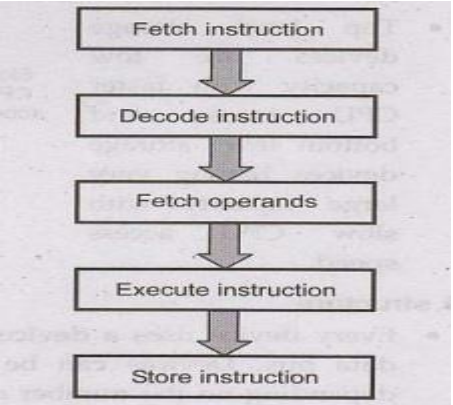


Fig. 1.1.3 Instruction execution cycle

• Fig 1.1.3 shows instruction execution cycle.

• The CPU "executes" the operation specified by the instruction on this data. The to CPU "stores" any results into a register or memory.

•The register used for instruction execution are as follows:

1. Memory Address Register (MAR): It specifies the address of memory location from which data or instruction is to be accessed (for read operation) or to best yr which the data is to be stored (for write operation).

2. Program Counter (PC): It keeps track of the instruction which is to be executed next, after the execution of an on-going instruction.

3. Instruction Register (IR): Here the instructions are loaded before their execution.

Interrupts

• **Definition:** It is an event external to the currently executing process that causes a change in the normal flow of instruction execution; usually generated by hardware not devices external to the CPU. They tell the CPU to stop its current activities and not execute the appropriate part of the operating system.

• Fig. 1.1.4 shows interrupts.

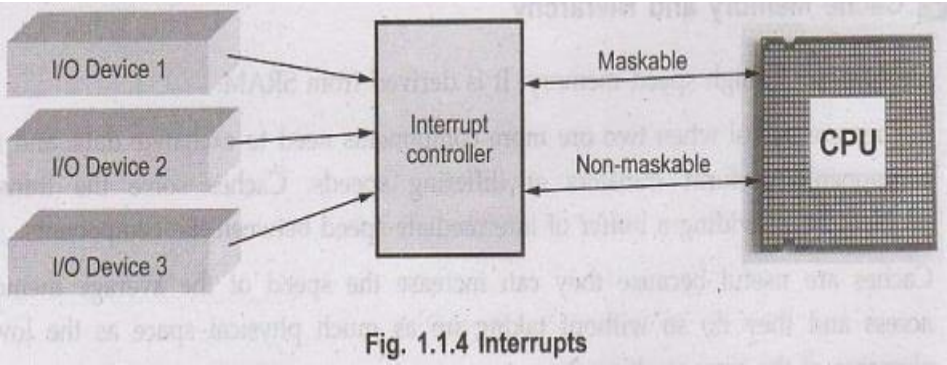


Fig. 1.1.4 Interrupts

• The CPU uses a table and the interrupt vector to find OS the code to execute in response to interrupts. When interrupt signaled, processor executes a routine called an interrupt handler to deal with the interrupt.

• Operating system may specify a set of instructions, called an interrupt handler, to be executed in response to each type of interrupt.

• Interrupt Service Routine (ISR) is the software code that is executed when the hardware requests interrupt. The design of the interrupt service routine requires careful consideration of many factors. Although interrupt handlers can create and use local variables, parameter passing between threads must be implemented using shared global memory variables.

• A private global variable can be used if an interrupt thread wishes to pass information to itself, e.g., from one interrupt instance to another. The execution of the main program is called the foreground thread, and the executions of the various interrupt service routines are called background threads.

• Interrupts are of three types: **Hardware, software and trap.**

• **Hardware Interrupts** are generated by hardware devices to signal that they need some attention from the OS. **Software Interrupts** are generated by programs when they want to request a system call to be performed by the operating system.

• Traps are generated by the CPU itself to indicate that some error or condition occurred for which assistance from the operating system is needed.

• Interrupt and trap numbers are defined by the hardware which is also responsible for calling the procedure in the kernel space. An interrupt handler is called in response to a signal from

another device while a trap handler is called in response to an instruction executed within the CPU.

• Synchronous interrupts occur when a process attempts to perform an illegal action, such as dividing by zero or referencing a protected memory location. Synchronous interrupt occurs due to software execution.

Cache Memory and Hierarchy

• Cache is small high speed memory. It is derived from SRAM.

• Caches are useful when two ore more components need to exchange data, and the components perform transfers at differing speeds. Caches solve the transfer problem by providing a buffer of intermediate speed between the components.

• Caches are useful because they can increase the speed of the average memory access and they do so without taking up as much physical space as the lower elements of the memory hierachy.

• Data is normally kept in storage system.. when it required, it is copied into a faster storage system i.e. cache memory for temporary basics.

• When user required a particular data, system check whether it is in the cache. If data found then, we use the data directly from the cache. It is not found then, use the data from the source.

• Internal programmable registers, such as index registers, provide a high-speed cache for main memory. The programmer implements the registor-allocation and register-replacement algorithms to decide which information to keep in registers and which to keep in main memory.

• If the fast device finds the data it needs in the cache, it need not wait for the 1er slower device. The data in the cache must be kept consistent with the data in the components.

• If a component has data value change, and the datum is also in the cache, the cache must also be updated. This is especially a problem on multiprocessor systems where more than one process may be accessing a datum.

• A component may be eliminated by an equal sized cache, but only if the cache and the component have equivalent state-saving capacity and the cache is affordable, because aster storage tends to be more expensive.

• Unfortunately, cache also introduce an additional level of complexity (coherency and consistency assurance).We also incur an economic and space penalty when we add a cache.

• Making a cache as large as a disk would be ineffective because it would be too costly, the immense size would slow it down and a cache is generally a volatile memory, while we want data on disks to be persistent..

• It holds data for temporary purpose to reduce the time required to service I/O requests from the host. Fig. 1.1.5 shows memory hierarchy.

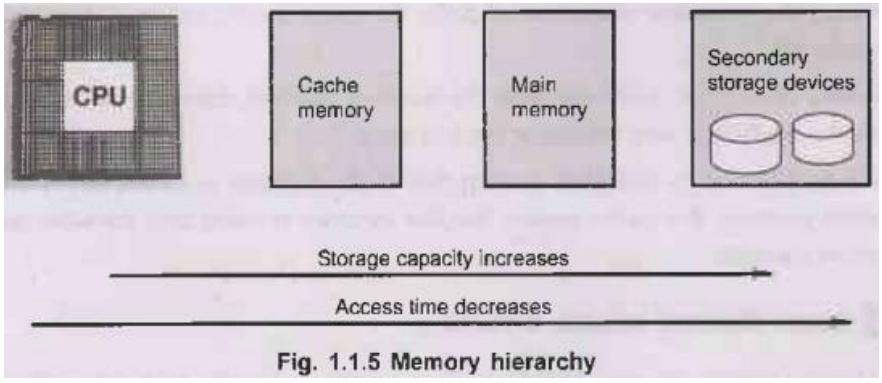


Fig. 1.1.5 Memory hierarchy

• Cache in CPU unit is referred as Level1 cache (L1 cache) and cache stored in a chip next to CPU is termed as Level2 cache (L2 cache) and it resides on the motherboard.

• The function of cache is to act as a buffer between a relatively fast device and a relatively slow one.

• Small unit of allocation in cache is page. Cache is arranged into slots or pages. It uses two components data store and tag RAM. The actual data is stored in a

different part of the cache, called the data store. The values stored in the tag RAM determine whether a cache lookup results in a hit or a miss.

• The size of the tag RAM determines what range of main memory can be cached. Tag RAM is a small piece of SRAM that stores the addresses of the data that is stored in the SRAM

• A cache address can be specified simply by index and offset. The tag is kept to allow the cache to translate from a cache address to a unique CPU address.

• A cache hit means that the CPU tried to access an address, and a matching cache block was available in cache. So, the cache did not need to access RAM. In a cache miss, the CPU tries to access an address, and there is no matching cache block. So, the cache is forced to access RAM. of

• Cache includes tags to identify which block of main memory is in each cache slot.

• Every cache block has associated with it at least the modify and valid bits, and a tag address. The valid bit says if the cache block is used or is unused. The modify bit makes sense only if the valid bit is set. The modify bit says whether the data in the cache block is different from RAM or is the same as RAM.

• The size of a page is dependent on the size of the cache and how the cache is organized. A cache page is broken into smaller pieces, each called a cache line. The size of a cache line is determined by both the processor and the cache design. .

• When the processor starts a read cycle, the cache checks to see if that address is a cache hit.

• **Cache Hit:** If the cache contains the memory location, then the cache will respond to the read cycle and terminate the bus cycle.

• **Cache Miss:** It is reference to item that is not resident in cache, but is resident in main memory. For cache misses, the fast memory is cache and the slow memory is main memory.

Direct Memory Access Structure

• Direct Memory Access (DMA) is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the core processing unit or memory in a computer. DMA is one of the faster types of synchronization mechanisms.

• Without DMA, the processor is responsible for the physical movement of data between main memory and a device. A special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This approach is called direct memory access.

• Fig. 1.1.6 shows simplified diagram of DMA controller.

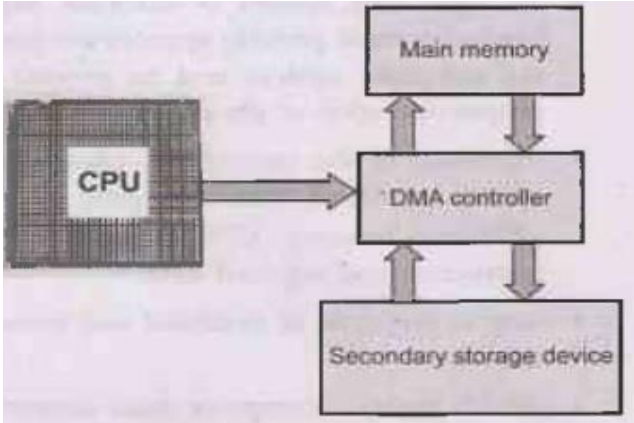


Fig. 1.1.6 DMA structure

• The DMA controller uses hold request (HOLD) and hold girl acknowledge (HOLD A) signals to ask the CPU to stop driving the address, data and control buses so that the DMA controller can drive them to carry out a transfer.

• A DMA controller implements direct memory access in a computer system. It connects directly to the I/O device at one end and to the system buses at the other end. It also interacts with the CPU, both via the system buses and two new direct connections.

• Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention. Only one interrupt is generated per block, rather than the one interrupt per byte.

• The DMA controller may either stop the CPU and access the memory or use the bus while the CPU is not using it

Operating System Generations

Operating systems have been evolving over the years. We can categorise this evaluation based on different generations which is briefed below:

0^{th} Generation

The term 0^{th} generation is used to refer to the period of development of computing when Charles Babbage invented the Analytical Engine and later John Atanasoff created a computer in 1940. The hardware component technology of this period was electronic vacuum tubes. There was no Operating System available for this generation computer and computer programs were written in machine language. This computers in this generation were inefficient and dependent on the varying competencies of the individual programmer as operators.

First Generation (1951-1956)

The first generation marked the beginning of commercial computing including the introduction of Eckert and Mauchly's UNIVAC I in early 1951, and a bit later, the IBM 701.

System operation was performed with the help of expert operators and without the benefit of an operating system for a time though programs began to be written in higher level, procedure-oriented languages, and thus the operator's routine expanded. Later mono-programmed operating system was developed, which eliminated some of the human intervention in running job and provided programmers with a number of desirable functions. These systems still continued to operate under the control of a human operator who used to follow a number of steps to execute a program. Programming language like FORTRAN was developed by John W. Backus in 1956.

Second Generation (1956-1964)

The second generation of computer hardware was most notably characterised by transistors replacing vacuum tubes as the hardware component technology. The first operating system GMOS was developed by the IBM computer. GMOS was based on single stream batch processing system, because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine. Operating

system is cleaned after completing one job and then continues to read and initiates the next job in punch card.

Researchers began to experiment with multiprogramming and multiprocessing in their computing services called the time-sharing system. A noteworthy example is the Compatible Time Sharing System (CTSS), developed at MIT during the early 1960s.

Third Generation (1964-1979)

The third generation officially began in April 1964 with IBM's announcement of its System/360 family of computers. Hardware technology began to use integrated circuits (ICs) which yielded significant advantages in both speed and economy.

Operating system development continued with the introduction and widespread adoption of multiprogramming. The idea of taking fuller advantage of the computer's data channel I/O capabilities continued to develop.

Another progress which leads to developing of personal computers in fourth generation is a new development of minicomputers with DEC PDP-1. The third generation was an exciting time, indeed, for the development of both computer hardware and the accompanying operating system.

Fourth Generation (1979 – Present)

The fourth generation is characterised by the appearance of the personal computer and the workstation. The component technology of the third generation, was replaced by very large scale integration (VLSI). Many Operating Systems which we are using today like Windows, Linux, MacOS etc developed in the fourth generation.

Main Objectives of the computer system

The objectives of an operating system (OS) are to make the computer system more efficient, convenient, and secure. It also manages the hardware resources of the computer.

Efficiency

- **Process management**
  Schedules processes to maximize CPU utilization and make the computer faster
- **Resource management**
  Manages the hardware resources of the computer, such as the CPU, memory, disk storage, and input/output devices
- **Fair sharing of resources**
  Provides efficient and fair sharing of resources between users and programs
  Convenience

- **User friendly**: Makes the computer system more interactive with a more convenient interface for the users
- **Easy access**: Provides easy access to users for using resources
  Security
- **Protection**: Helps in keeping the user data safe with help of protection like passwords
- **Unauthorized access**: Avoids unauthorized access to user data and programs
  Other objectives
- **Networking**: Provides networking capabilities, allowing the computer system to connect to other systems and devices over a network
- **Job accounting**: Keeps track of all the functions of a computer system, including memory, resources, and errors
- **Program execution**: Provides a convenient environment where users can run their programs

Operating System functions:

- Memory Management
- Processor Management
- Device Management
- File Management
- Network 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 accessed 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 part 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. This function is called **process scheduling**. 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 softwares and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Evolution of Operating system

Types of Operating System

1. Batch Operating System
2. Time-Sharing Operating System

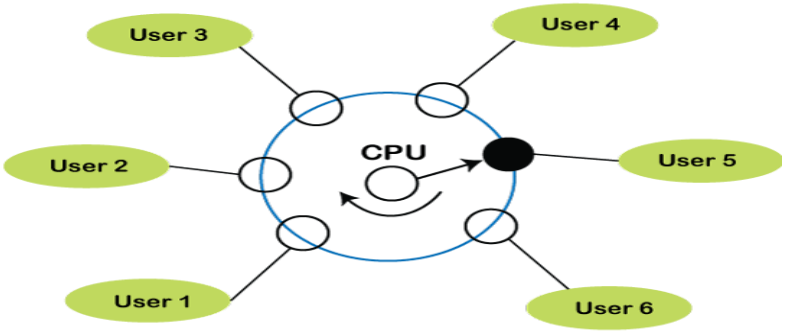
3. Embedded Operating System
4. Multiprogramming Operating System
5. Network Operating System
6. Distributed Operating System
7. Multiprocessing Operating System
8. Real-Time Operating System

Batch Operating System

In Batch Operating System, there is no direct interaction between user and computer. Therefore, the user needs to prepare jobs and save offline mode to punch card or paper tape or magnetic tape. After creating the jobs, hand it over to the computer operator; then the operator sort or creates the similar types of batches like B2, B3, and B4. Now, the computer operator submits batches into the CPU to execute the jobs one by one. After that, CPUs start executing jobs, and when all jobs are finished, the computer operator provides the output to the user.

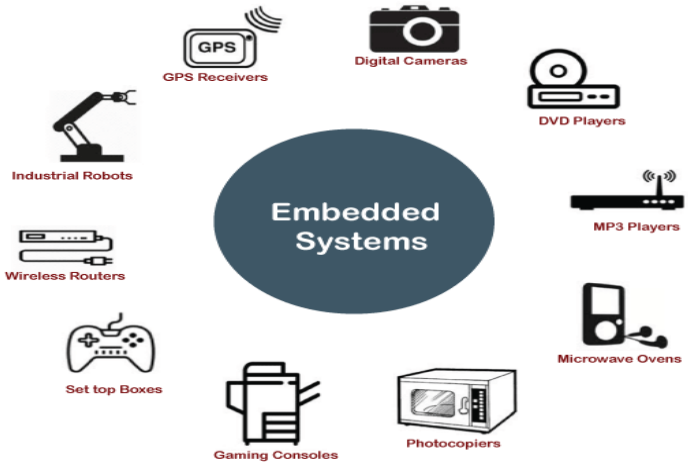
Time-Sharing Operating System

It is the type of operating system that allows us to connect many people located at different locations to share and use a specific system at a single time. The time-sharing operating system is the logical extension of the multiprogramming through which users can run multiple tasks concurrently. Furthermore, it provides each user his terminal for input or output that impacts the program or processor currently running on the system. It represents the CPU's time is shared between many user processes. Or, the processor's time that is shared between multiple users simultaneously termed as time-sharing.



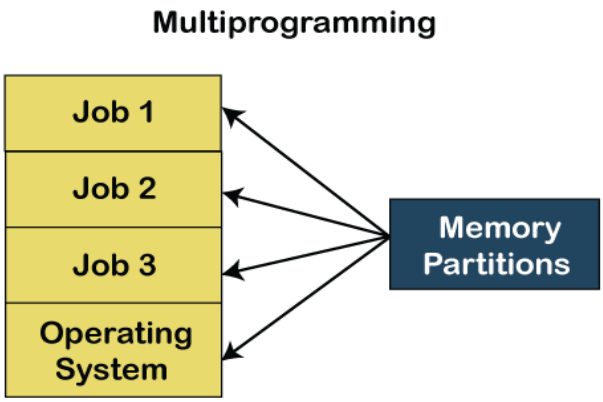
Embedded Operating System

The Embedded operating system is the specific purpose operating system used in the computer system's embedded hardware configuration. These operating systems are designed to work on dedicated devices like automated teller machines (ATMs), airplane systems, digital home assistants, and the internet of things (IoT) devices.



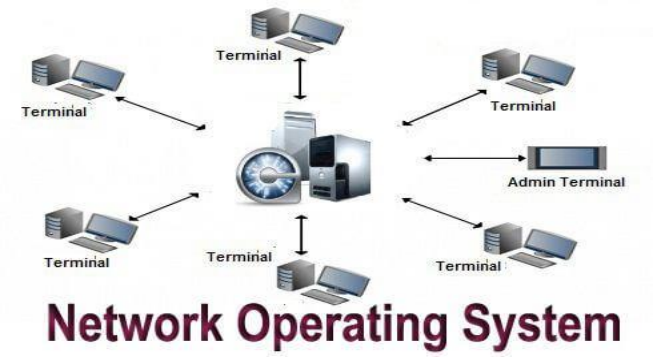
Multiprogramming Operating System

Due to the CPU's underutilization and the waiting for I/O resource till that CPU remains idle. It shows the improper use of system resources. Hence, the operating system introduces a new concept that is known as multiprogramming. A **multiprogramming operating system** refers to the concepts wherein two or more processes or programs activate simultaneously to execute the processes one after another by the same computer system. When a program is in run mode and uses CPU, another program or file uses I/O resources at the same time or waiting for another system resources to become available. It improves the use of system resources, thereby increasing system throughput. Such a system is known as a multiprogramming operating system.



Network Operating System

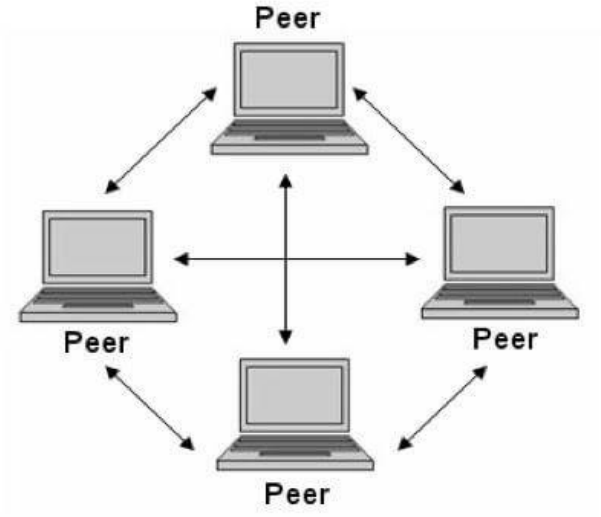
A network operating system is an important category of the operating system that operates on a server using network devices like a switch, router, or firewall to handle data, applications and other network resources. It provides connectivity among the autonomous operating system, called as a network operating system. The network operating system is also useful to share data, files, hardware devices and printer resources among multiple computers to communicate with each other.



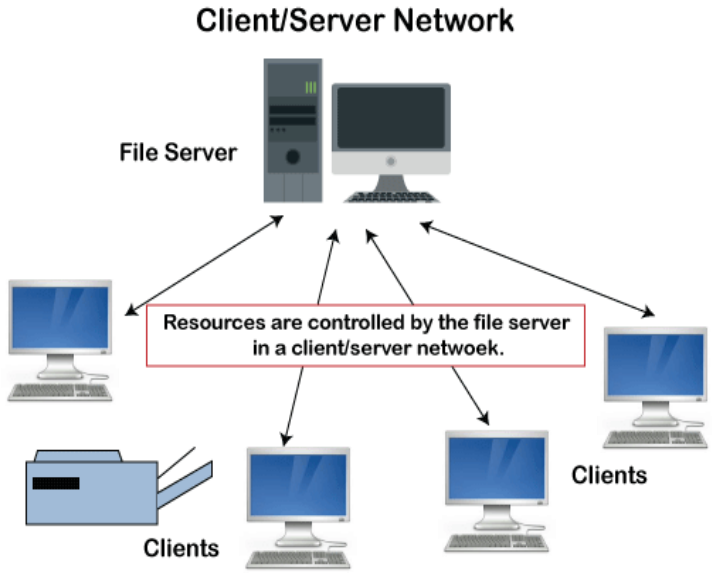
Types of network operating system

- **Peer-to-peer network operating system:** The type of network operating system allows users to share files, resources between two or more computer machines using a

LAN.



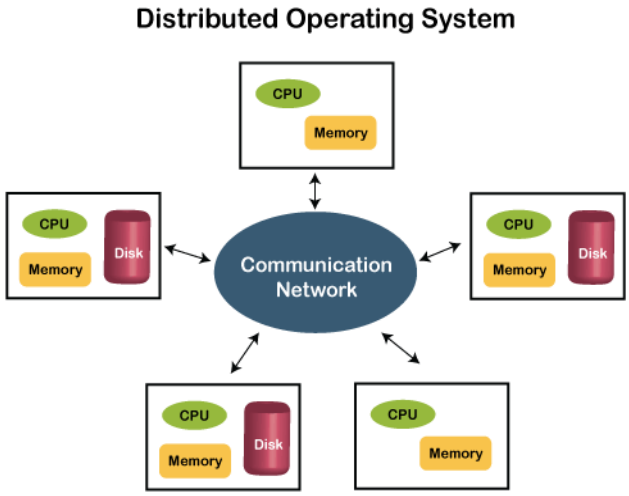
- **Client-Server network operating system:** It is the type of network operating system that allows the users to access resources, functions, and applications through a common server or center hub of the resources. The client workstation can access all resources that exist in the central hub of the network. Multiple clients can access and share different types of the resource over the network from different locations.



Distributed Operating system

A distributed operating system provides an environment in which multiple independent CPU or processor communicates with each other through physically separate computational nodes. Each node contains specific software that communicates with the global aggregate operating system. With the ease of a distributed system, the programmer or developer can easily access

any operating system and resource to execute the computational tasks and achieve a common goal. It is the extension of a network operating system that facilitates a high degree of connectivity to communicate with other users over the network.



Multiprocessing Operating System

It is the type of operating system that refers to using two or more central processing units (CPU) in a single computer system. However, these multiprocessor systems or parallel operating systems are used to increase the computer system's efficiency. With the use of a multiprocessor system, they share computer bus, clock, memory and input or output device for concurrent execution of process or program and resource management in the CPU.

Real-Time Operating System

A real-time operating system is an important type of operating system used to provide services and data processing resources for applications in which the time interval required to process & respond to input/output should be so small without any delay real-time system. For example, real-life situations governing an automatic car, traffic signal, nuclear reactor or an aircraft require an immediate response to complete tasks within a specified time delay. Hence, a real-time operating system must be fast and responsive for an embedded system, weapon system, robots, scientific research & experiments and various real-time objects.

Types of the real-time operating system:

- **Hard Real-Time System**
  These types of OS are used with those required to complete critical tasks within the defined time limit. If the response time is high, it is not accepted by the system or may face serious issues like a system failure. In a hard real-time system, the secondary storage is either limited or missing, so these system stored data in the ROM.
- **Soft Real-Time System**
  A soft real-time system is a less restrictive system that can accept software and hardware resources delays by the operating system. In a soft real-time system, a critical task prioritizes less important tasks, and that priority retains active until completion of the task. Also, a time limit is set for a specific job, which enables short time delays for further tasks that are acceptable. For example, computer audio or video, virtual reality, reservation system, projects like undersea, etc.

Operating System Structure

Overview

An operating system is a design that enables user application programs to communicate with the hardware of the machine. The operating system should be built with the utmost care because it is such a complicated structure and should be simple to use and modify. Partially developing the operating system is a simple approach to accomplish this. Each of these components needs to have distinct inputs, outputs, and functionalities.

This article discusses many sorts of structures that implement operating systems, as listed below, as well as how and why they work. It also defines the operating system structure.

- Simple Structure
- Monolithic Structure
- Layered Approach Structure
- Micro-Kernel Structure
- Exo-Kernel Structure

- Virtual Machines

Operating System Structure

Operating systems consist of many interacting parts, and the definition of the structural relationships among these parts is called OS structure. Such a structure ensures that it will be easy to develop, maintain, and customize the OS for particular applications. The clear structure lets the OS be built from manageable segments; each segment contributes to the overall system functionality.

The OS structure provides an idea of how different components are connected and integrated within the kernel, which is the central part of the OS. The kernel carries out sensitive tasks such as memory management, process scheduling, and interaction with hardware. A look into different OS structures and their relative advantages and disadvantages is presented next.

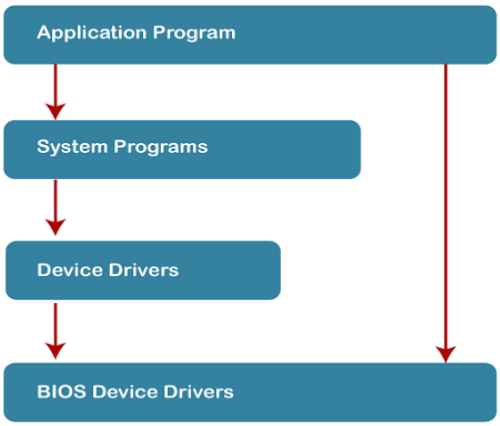
1. Simple Structure

It is the most straightforward operating system structure, but it lacks definition and is only appropriate for usage with tiny and restricted systems. Since the interfaces and degrees of functionality in this structure are clearly defined, programs are able to access I/O routines, which may result in unauthorized access to I/O procedures.

This organizational structure is used by the MS-DOS operating system:

- There are four layers that make up the MS-DOS operating system, and each has its own set of features.
- These layers include ROM BIOS device drivers, MS-DOS device drivers, application programs, and system programs.
- The MS-DOS operating system benefits from layering because each level can be defined independently and, when necessary, can interact with one another.
- If the system is built in layers, it will be simpler to design, manage, and update. Because of this, simple structures can be used to build constrained systems that are less complex.
- When a user program fails, the operating system as whole crashes.
- Because MS-DOS systems have a low level of abstraction, programs and I/O procedures are visible to end users, giving them the potential for unwanted access.

The following figure illustrates layering in simple structure:



Advantages of Simple Structure:

- It is easy to develop as it contains only a few interfaces and levels.
- It executes faster because there are fewer layers between the hardware and applications.
-
- Direct access to hardware provides faster operations.
- Minimal consumption of system resources as the structure is very simple.
- Light in weight hence suitable for small or even embedded systems.

Disadvantages of Simple Structure:

- As these components are not distinctly separated, making any change or upgrade in the system is considerably harder to achieve.
- All things are connected, and therefore, debugging is quite hectic; finding the error is very hard.
- Due to the lack of data abstraction, a security vulnerability is a strong possibility.
- It does not scale well to larger, more complex systems. Adding new features is particularly challenging in this architecture.
- One failure in one part might bring down the whole system.

---

2. MONOLITHIC STRUCTURE

The monolithic operating system controls all aspects of the operating system's operation, including file management, memory management, device management, and operational operations.

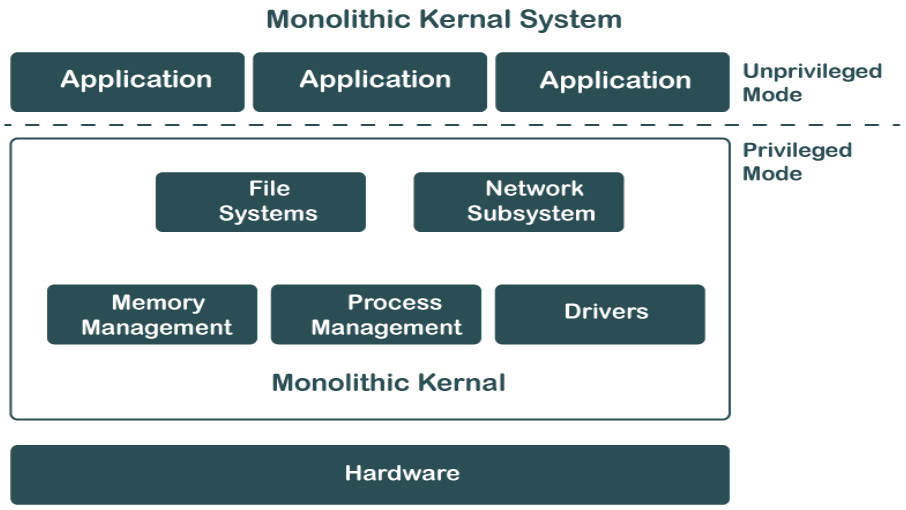
The core of an operating system for computers is called the kernel (OS). All other System components are provided with fundamental services by the kernel.

The operating system and the hardware use it as their main interface. When an operating system is built into a single piece of hardware, such as a keyboard or mouse, the kernel can directly access all of its resources.

The monolithic operating system is often referred to as the monolithic kernel. Multiple programming techniques such as batch processing and time-sharing increase a processor's usability. Working on top of the operating system and under complete command of all hardware, the monolithic kernel performs the role of a virtual computer.

This is an old operating system that was used in banks to carry out simple tasks like batch processing and time-sharing, which allows numerous users at different terminals to access the Operating System.

The following diagram represents the monolithic structure:



Advantages of Monolithic Structure:

- Because everything is packed into one big kernel, there is no need for multi-layer complexities. Hence, the design and bringing into practice of such a system is relatively easy.
- As long as all the components run in the same memory space, there is no need to have inter-process communication or to perform context switching between layers; thus, execution is faster.
- Direct access to hardware and little additional-abstraction overhead result in high performance.
- Development can thus be easier as developers have to deal with one codebase only, and no interactions among various layers or modules need to be considered.

Disadvantages of Monolithic Structure:

- Fault in any single component has the potential to bring down the whole system since there is no isolation between different parts.
- This can be a hard and unsafe task, as changes in one part of the system may accidentally filter out into another part.
- Since the components are more interdependent, debugging is also more complex and always requires heavy testing and testing-related activities.
- This is an issue where scaling up or adjusting to the latest trends in a system involves significant changes in the kernel.
- Consequently, it will be cumbersome to update or upgrade the system because it involves a large codebase in a monolithic form where changes may have widespread effects.

---

3. LAYERED STRUCTURE

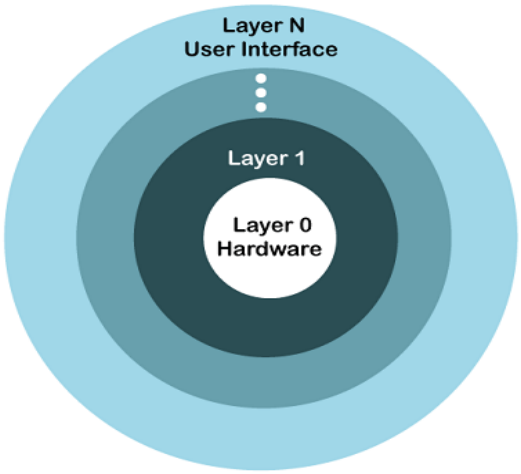
The OS is separated into layers or levels in this kind of arrangement. Layer 0 (the lowest layer) contains the hardware, and layer 1 (the highest layer) contains the user interface (layer N).

- The model, in this respect, uses the functions' availability through the layer below it. For example, memory management may have to rely on process scheduling, while higher layers use it here with respect to user interaction.

- This approach also introduces abstraction between layers, whereby the system is also easier to manage and debug. In case any issue arises, testing and debugging can be performed at lower layers before going onto higher layers

The functionalities of each layer are separated in this method, and abstraction is also an option. Because layered structures are hierarchical, debugging is simpler, therefore all lower-level layers are debugged before the upper layer is examined. As a result, the present layer alone has to be reviewed since all the lower layers have already been examined.

The image below shows how OS is organized into layers:



Advantages of Layered Structure:

- Development, maintenance, and understanding are much easier, as every layer is assigned a specific role.
- It can isolate problems in individual layers because testing and debugging would be less demanding.
- Sensitive operations can be confined to specific layers; thus, unauthorized access can be minimized to such layers because the concerns have been clearly separated.
- Modifying or updating one layer will not affect other independent layers directly; thus, upgrading or fixing could be done more easily.

- With layers, basic operations can be entrusted to lower layers, and hence, the layers above need not be concerned with the minute details, thereby reducing the complexity of the overall management.

Disadvantages of Layered Structure:

- The layered approach adds delays due to the fact that data and commands must pass through several layers, thus slowing down the system in general.
- Designing a properly working layered structure requires forethought and iron discipline for proper interaction between different layers.
- Layers are generally dependent on lower layers; thus skipping layers and directly accessing the lower-level functionalities becomes harder, hence reducing flexibility in many systems.
- System performance optimization becomes challenging due to the extra abstractions introduced by the layers, which harden the tuning of individual parts of the system to make them run more effectively.
- Unless the lower-layer design is lousy or inefficient, the whole system will suffer since the higher layer depends on them for functionality.

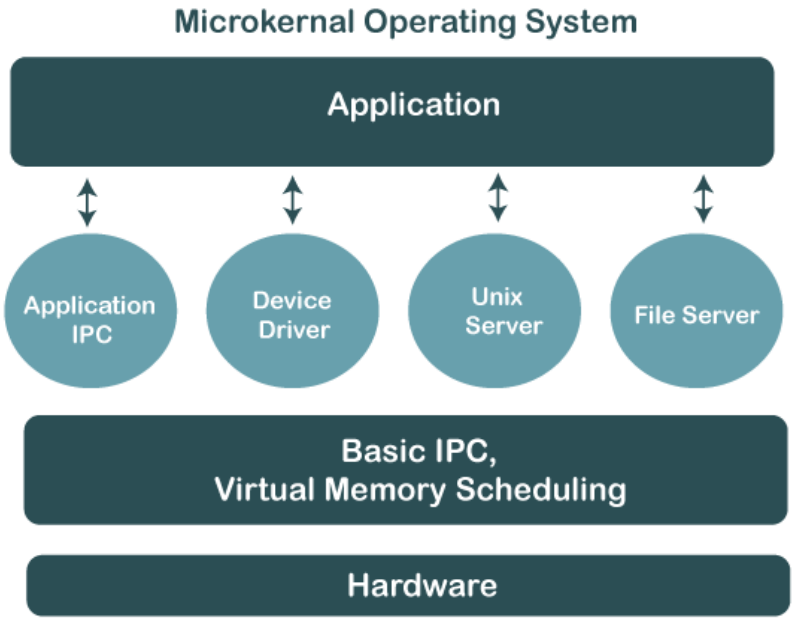
4.MICRO-KERNEL STRUCTURE

The operating system is created using a micro-kernel framework that strips the kernel of any unnecessary parts. Systems and user applications are used to implement these optional kernel components. So, Micro-Kernels is the name given to these systems that have been developed.

The **Micro-Kernel Structure** minimizes the functions handled by the kernel, delegating most services to user-level applications. The micro-kernel itself only handles core functions, such as memory management, process scheduling, and basic communication between processes.

- By stripping away unnecessary services from the kernel, the system becomes more secure and reliable. If a service crashes, it doesn't affect the entire system.
- Micro-kernels are common in modern OS architectures like macOS and QNX, as they allow for easier updates and better system resilience.

The image below shows the Micro-Kernel Operating System Structure:



Advantages of Micro-Kernel Structure:

- The just necessary services running in the kernel ensure that user service failures don't crash the whole system.
- It isolates the services from the kernel, thus minimizing the possibility of compromising the system since every service operates independently of the other.
- Smaller kernels are easier to operate and alter. Hence, updating and bug fixes are less challenging.
- Services can be added or removed without affecting the kernel. So, this gives more flexibility while designing the system.
- Due to the small size of the micro-kernel, it can be ported on different hardware platforms with minimal modification.

Disadvantages of Micro-Kernel Structure:

- The inter-process communication between kernel and user services introduces additional overhead and hence is slower as compared to the monolithic approach.
- In a micro-kernel, the separation of services must be done very carefully, which may complicate the initial development process.
- Context switches are more frequent between user space and kernel space in this approach; hence, performance would be slower in highly loaded conditions.
- Due to the complexity and possibly extra performance trade-offs, few operating systems have been built using the micro-kernel architecture.

- Modular requires more planning and testing in detail; hence, the development cycle may be longer.

5. EXOKERNEL

An operating system called Exokernel was created at MIT with the goal of offering application-level management of hardware resources. The exokernel architecture's goal is to enable application-specific customization by separating resource management from protection. Exokernel size tends to be minimal due to its limited operability.

Because the OS sits between the programs and the actual hardware, it will always have an effect on the functionality, performance, and breadth of the apps that are developed on it. By rejecting the idea that an operating system must offer abstractions upon which to base applications, the exokernel operating system makes an effort to solve this issue. The goal is to give developers as few restriction on the use of abstractions as possible while yet allowing them the freedom to do so when necessary. Because of the way the exokernel architecture is designed, a single tiny kernel is responsible for moving all hardware abstractions into unreliable libraries known as library operating systems. Exokernels differ from micro- and monolithic kernels in that their primary objective is to prevent forced abstraction.

Exokernel operating systems have a number of features, including:

- Enhanced application control support.
- Splits management and security apart.
- A secure transfer of abstractions is made to an unreliable library operating system.
- Brings up a low-level interface.
- Operating systems for libraries provide compatibility and portability.

Advantages of Exokernel Structure:

- Applications directly interface with hardware to realize high power.
- It provides fine-grained resource allocation and revocation for efficient usage.
- Testing and development of new operating systems with low-level control become easy.
- Each application can have its memory management; thus, optimization would be tailored.

- It removes extra kernel features; thus, it makes the system lean and faster.
- Because of its minimalistic approach, it is easily adaptable to different hardware platforms.

Disadvantages of Exokernel Structure:

- Developing and maintaining such a complicated interface would be challenging.
- The abstractions are not enforced, which might make the different applications inconsistent with each other.
- It makes the developers responsible for dealing with resource management, which enhances the difficulty level of it.
- Lack of strong abstractions with direct access to hardware may result in security vulnerabilities.
- Debugging is more difficult since most of the system is on a low level of abstraction.
- This means that older applications, which rely on lower-level abstractions, will not function well since the latter abstract higher-level operating system abstractions.

---

6. VIRTUAL MACHINES (VMs)

The hardware of our personal computer, including the CPU, disc drives, RAM, and NIC (Network Interface Card), is abstracted by a virtual machine into a variety of various execution contexts based on our needs, giving us the impression that each execution environment is a separate computer. A virtual box is an example of it.

Using CPU scheduling and virtual memory techniques, an operating system allows us to execute multiple processes simultaneously while giving the impression that each one is using a separate processor and virtual memory. System calls and a file system are examples of extra functionalities that a process can have that the hardware is unable to give. Instead of offering these extra features, the virtual machine method just offers an interface that is similar to that of the most fundamental hardware. A virtual duplicate of the computer system underneath is made available to each process.

We can develop a virtual machine for a variety of reasons, all of which are fundamentally connected to the capacity to share the same underlying hardware while concurrently supporting various execution environments, i.e., various operating systems.

Disk systems are the fundamental problem with the virtual machine technique. If the actual machine only has three-disc drives but needs to host seven virtual machines, let's imagine that. It is obvious that it is impossible to assign a disc drive to every virtual machine because the program that creates virtual machines would require a sizable amount of disc space in order to offer virtual memory and spooling. The provision of virtual discs is the solution.

The result is that users get their own virtual machines. They can then use any of the operating systems or software programs that are installed on the machine below. Virtual machine software is concerned with programming numerous virtual machines simultaneously into a physical machine; it is not required to take into account any user-support software. With this configuration, it may be possible to break the challenge of building an interactive system for several users into two manageable chunks.

Advantages of Virtual Machines:

- Each VM is in itself independent, which means any failure or security consequences that take place will not affect the other virtual machines.
- One can have a number of operating systems running on one hardware platform.
- Cloning, backups, and restoration of virtual machines are pretty easy; hence, options galore as far as data recovery is concerned.
- Different operating systems, such as Windows, Linux, etc., can be used with one machine itself.
- They are ideal for testing software updates or any OS in an isolated environment without incurring changes anywhere in the host system.
- They can be easily moved between different physical machines with very little effort.

Disadvantages of Virtual Machines:

- Running more than one virtual machine on the same host may decrease its performance due to the sharing of the resources.
- VMs are CPU, memory-intensive, and storage-intensive applications; this will put a high load on host systems.
- Managing many virtual machines requires further advanced tools and knowledge.
- It cannot utilize the hardware as effectively, as would a physical machine, and hence it affects performance-related applications.

- Uncontrolled creation can lead to disorganization, inefficient use of resources, and security risks.
- Vulnerabilities in either the hypervisor or VM management software may have the potential to expose all VMs to security threats.

Operating System Services

An operating system is software that acts as an intermediary between the user and computer hardware. It is a program with the help of which we are able to run various applications. It is the one program that is running all the time. Every computer must have an operating system to smoothly execute other programs.

The OS coordinates the use of the hardware and application programs for various users. It provides a platform for other application programs to work. The operating system is a set of special programs that run on a computer system that allows it to work properly. It controls input-output devices, execution of programs, managing files, etc.

Services of Operating System

- Program execution
- Input Output Operations
- Communication between Process
- File Management
- Memory Management
- Process Management
- Security and Privacy
- Resource Management
- User Interface
- Networking
- Error handling
- Time Management

Program Execution

It is the Operating System that manages how a program is going to be executed. It loads the program into the memory after which it is executed. The order in which they are executed depends on the CPU Scheduling Algorithms. A few are FCFS, SJF, etc. When the program is in execution, the Operating System also handles deadlock i.e. no two processes come for execution at the same time. The Operating System is responsible for the smooth execution of

both user and system programs. The Operating System utilizes various resources available for the efficient running of all types of functionalities.

Input Output Operations

Operating System manages the input-output operations and establishes communication between the user and device drivers. Device drivers are software that is associated with hardware that is being managed by the OS so that the sync between the devices works properly. It also provides access to input-output devices to a program when needed.

Communication Between Processes

The Operating system manages the communication between processes. Communication between processes includes data transfer among them. If the processes are not on the same computer but connected through a computer network, then also their communication is managed by the Operating System itself.

File Management

The operating system helps in managing files also. If a program needs access to a file, it is the operating system that grants access. These permissions include read-only, read-write, etc. It also provides a platform for the user to create, and delete files. The Operating System is responsible for making decisions regarding the storage of all types of data or files, i.e, floppy disk/hard disk/pen drive, etc. The Operating System decides how the data should be manipulated and stored.

Memory Management

Let's understand memory management by OS in simple way. Imagine a cricket team with limited number of player . The team manager (OS) decide whether the upcoming player will be in playing 11 ,playing 15 or will not be included in team , based on his performance . In the same way, OS first check whether the upcoming program fulfil all requirement to get memory space or not ,if all things good, it checks how much memory space will be sufficient for program and then load the program into memory at certain location. And thus , it prevents program from using unnecessary memory.

Process Management

Let's understand the process management in unique way. Imagine, our kitchen stove as the (CPU) where all cooking(execution) is really happen and chef as the (OS) who uses kitchen-stove(CPU) to cook different dishes(program). The chef(OS) has to cook different dishes(programs) so he ensure that any particular dish(program) does not take long time(unnecessary time) and all dishes(programs) gets a chance to cooked(execution) .The

chef(OS) basically scheduled time for all dishes(programs) to run kitchen(all the system) smoothly and thus cooked(execute) all the different dishes(programs) efficiently.

Security and Privacy

- **Security :** OS keep our computer safe from an unauthorized user by adding security layer to it. Basically, Security is nothing but just a layer of protection which protect computer from bad guys like viruses and hackers. OS provide us defenses like firewalls and anti-virus software and ensure good safety of computer and personal information.
- **Privacy :** OS give us facility to keep our essential information hidden like having a lock on our door, where only you can enter and other are not allowed . Basically , it respect our secrets and provide us facility to keep it safe.

Resource Management

System resources are shared between various processes. It is the Operating system that manages resource sharing. It also manages the CPU time among processes using CPU Scheduling Algorithms. It also helps in the memory management of the system. It also controls input-output devices. The OS also ensures the proper use of all the resources available by deciding which resource to be used by whom.

User Interface

User interface is essential and all operating systems provide it. Users either interacts with the operating system through the command-line interface or graphical user interface or GUI. The command interpreter executes the next user-specified command.

A GUI offers the user a mouse-based window and menu system as an interface.

Networking

This service enables communication between devices on a network, such as connecting to the internet, sending and receiving data packets, and managing network connections.

Error Handling

The Operating System also handles the error occurring in the CPU, in Input-Output devices, etc. It also ensures that an error does not occur frequently and fixes the errors. It also prevents the process from coming to a deadlock. It also looks for any type of error or bugs that can occur while any task. The well-secured OS sometimes also acts as a countermeasure for preventing any sort of breach of the Computer System from any external source and probably handling them.

Time Management

Imagine traffic light as (OS), which indicates all the cars(programs) whether it should be stop(red)=>(simple queue), start(yellow)=>(ready queue),move(green)=>(under execution)

and this light (control) changes after a certain interval of time at each side of the road(computer system) so that the cars(program) from all side of road move smoothly without traffic.

User and Operating System Interface

• Various methods are used to interface with the operating system. They are command-line interface, Graphical User Interface and Touch-Screen Interface.

Command Interpreters

• A command interpreter is a system software that understands and executes commands that are entered interactively by a human or from another program. Command interpreter is an important part of any operating system. It provides an interface between the user and the computer.

• Command interpreter is also called command-line interface.

• Most operating systems, including Linux, UNIX and Windows, treat the command interpreter as a special program that is running when a process is initiated or when a user first logs on.

• A command interpreter is often also called a command shell or simply a shell. A command shell also usually provides a set of programs or utilities which allows ad users to perform file management functions.

• The shell utility provides a user interface to many system services. For example, user requests such as listing file names in a directory, running a program, logging out, may all be handled by the shell. The shell may invoke other utilities to actually do the work; for example, directory file listing is sometimes a utility program itself.

• The command interpreter itself contains the code to execute the command. For example, a command to delete a file may cause the command interpreter to jump to a section of its code that sets up the parameters and makes the appropriate system call.

• User may choose among several different shells, including the C shell, Bourne-Again shell, Korn shell, and others. Third-party shells and free user-written shells are also available. A very popular shell on most commercial variants of Unix is the Korn shell.

• Command line is text based interface of the Linux system. Terminal is another name in Linux system. User can communicate with system is by using command. You can combine more than one command and get output.

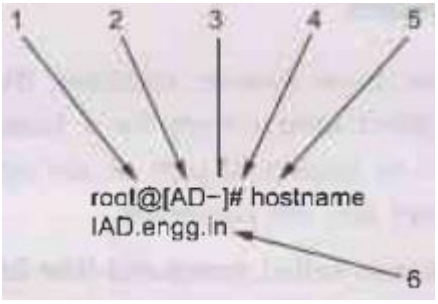
• Commands are executed at the prompt. Command line interface gives the user more control and options. There are many varieties of Linux, but almost all of them use similar commands that can be entered from a command-line interface terminal.

• The syntax of command line is as follows:

#command options

• Example :

[root@IAD~]# hostname



Here we will explain all of above arrows one by one.

1. Logged in user name, as now currently we are logged in with root user.

2. After the @ sign we have hostname of our Linux.

3. This is current working directory, "~" it is user home directory sign in Linux.

4. "#" hash sign or number sign means we are logged in with super user such as "root" it will be changed in case if we are logged in with normal Linux user e.g "$"

5. Just for example we give "hostname" command to shell, hostname is used to know the fully qualified hostname of your Linux machine.

6. This is output of our previously entered command.

Graphical User Interface

• GUI provides a graphical interface for the user to interact with the computer. It is the common user interface that includes graphical representation like buttons and Marie icons and communication can be performed by interacting with these icons rather than the usual text-based or command-based communication.

• The GUI actually translates user language, which comprises simple one-line commands, single click and double clicks to machine language or assembly language. The machine

understands machine language and hence the machine responds to the task initiated, which is translated to use language and communicated to the user via GUI.

• The first commercially available GUI was called "PARC," and was developed by Xerox for its 8010 Information System, released in 1981.

• The actions in a GUI are usually performed through direct manipulation of graphical elements like buttons and icons. Communication can be performed by interacting with these icons rather than the usual text-based or command-based

communication.

• Graphical user interface applications are self-descriptive, feedback is typically immediate and visual cues encourage and steer discoverability

Touch-Screen Interface

• The first touch screen phone was launched in 1992 by IBM and was named IBM Simon. iPhone was the latest invention of the touch screen phone which gets its popularity in no time.

• Whether it is touch screen microwave, touch screen car LCD, touch screen monitor, touch screen ATMs and others. The interface of touch screen devices is made in such a way that it is easy for people to use them.

• Some touch screen interface is fully controlled by the touch screen while others are partially controlled by the screen and another function is controlled by the physical keyboard.

• A touch user interface is a type of interaction between computer-based device. This interaction is by doing a physical touch on the screen and the computer responds to this touch interaction.

System Calls

• System calls provide the interface between a running program and the operating system. Any single CPU computer can execute only one instruction at a time. If a process is running a user program in user mode and needs a system service, such UTD as reading a data from a file, it has to execute a instruction to transfer control to the operating system.

• Operating system provides services and system call provides interface to these services. System call is written in language C and C++ as routines. System calls are performed in a series of steps.

• System call is a technique by which a program executing in user mode can request the kernel's service.

•An application programmer interface is a function definition that specifies how to

obtain a given service.

• Fig. 1.12.1 shows the working of system call.

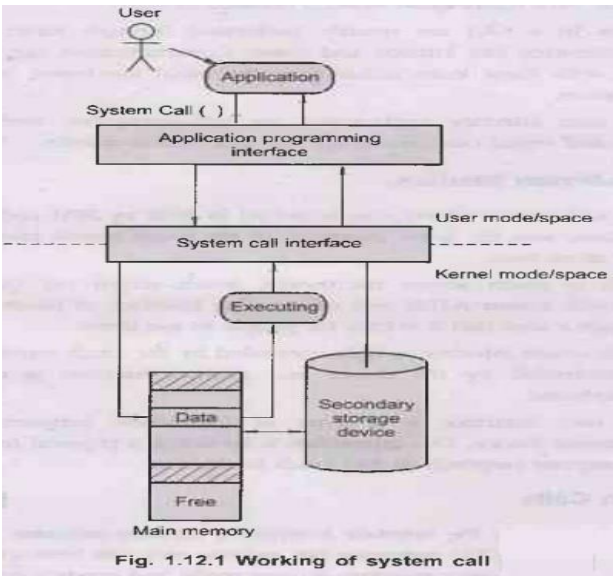


Fig. 1.12.1 Working of system call

• When application program calls the stub, trap instruction is executed and CPU switches to supervisor mode. Each system call contains its identification number.

• OS maintains the table of system call number. Operating system executes the system call using that number.

• When the function completes, it switches the processor to user mode and then returns control to the user process.

A system call is an explicit request to the kernel mode via a software interrupt. When user mode process invokes a system call, the CPU switches to kernel mode and starts the execution of the kernel function.

• Making a system call is like making a special kind of procedure call. Only system call enters the kernel and procedure call does not enter into the kernel.

• Kernel implements many different types of system calls. The user mode process must pass a parameter called the system call number to identify the required system call. All system calls return an integer value. In the kernel, positive or 0 values denote a successful termination of the system call and negative values denote an error condition.

• An API does not necessarily correspond to a specific system calls.

1. API could offer its services directly in user mode.

2. Single API function could make several system calls.

3. Several API functions could make the same system call.

• API supports a set of functions for application programmer. It includes passing parameter to each function and return values. API used by application programmer are as follows:

1. Windows system: win32 API

2. POSIX system : POSIX API

3. Virtual machine: Java API.

• User function uses trap instruction for using kernel services. Application programmer uses ordinary procedure call. Operating system provides a library of user functions with name corresponding to each actual system call. Each of these stub function contains a trap to the operating system function.

• Trap is also called system call interface.

• Three general methods are used to pass parameters to the operating system.

a. Pass parameters in registers

b. Registers pass starting addresses of blocks of parameters

C. Parameters can be placed, or pushed, onto the stack by the program, and

popped off the stack by the OS

• Fig. 1.12.2 shows passing of parameters as a table

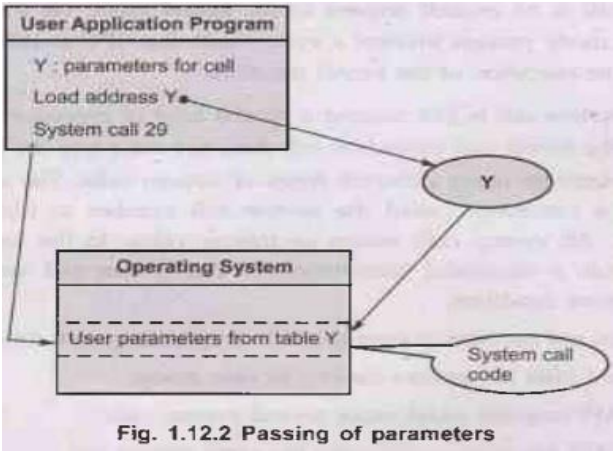


Fig. 1.12.2 Passing of parameters

Classification of System Call

• System call is divided into following types:

1. File management 2. Process management

3. Interprocess communication 4. I/O device management

5. Information processing and maintenance

1. File management

• File management system calls are create file, delete file, open file, close file, read file, write file, get and set file attribute.

• User can create a file using create() system call. File name with attributes are required for creating and deleting a file through system call. After creating a file, user can performs various operations on the file.

• Read, write, reposition are the operation performed on the file. File is closed after finished using. Same type of operation is performed on directory.

• Every file has file attributes. File attributes includes name of file, type of file, accounting information etc. To perform any operation on the file, set file attribute and get file attribute executed to check the attributes.

2. Process management

• System calls for process management are create, terminate, load, execute, abort, set and get process attributes. Other system call for process management is wait for time, wait event, allocate and free memory.

• In some situation user want to terminate the currently running process abnormally then system call used. Other reasons for abnormal process termination are error message generated, memory dump, error trap.

• Operating system provides debugging facility to determine the problem of dump. Dump is written to secondary storage disk.

• Debugger is a one type of system program. It provides facility for finding and correcting bug.

3. Interprocess communication

• Pipe, socket, message passing and shared memory are used for interprocess communication. Send message, receive message, create and delete connection, attach remote device are the system calls used in interprocess communication. •Shared memory: A process uses memory

for communication. One process will create a memory portion which other processes can access. A shared segment can be attached multiple times by the same process. Shared memory is the fastest form of IPC because data does not need to be copied between processes.

• A socket is a bidirectional communication device that can be used to communicate with another process on the same machine or with a process running on other machine.

• Message passing: Two processes communicate with each other by passing messages. Message passing is direct and indirect communication. Indirect communication uses mailbox for sending receiving message from other process.

4. I/O device management

• System calls for device management are request () device, release ( ) device, get of nail and set device attributes, read, write etc.

• Process needs several resources in its life time. When process request for resources, request is granted if it is free otherwise request is rejected. Once request is granted, control is transfer to the process.

5. Information processing and maintenance

• System calls for this category is set time and date, get time and date, get system data, get system data, get and set process, file and device.

• Most of the operating system provides system call for set and get the time and

date.

• This type of system call is used for transferring information from user to operating system and vice versa.

System Programs

• Modern operating system consists of a collection of system programs. System program that provides an application programming environment on top of the hardware. Some of them are simply user interfaces to system calls. They can be divided into these categories:

1. File management 2. Status information

3. File modification 4. Programming language support

5. Program loading and execution 6. Communications

• File management programs are used to create, delete, copy, rename, list, dump on files and directory. Status information system programs covers the date, time, disk in space, available memory and users. All this information is formatted and to displayed on output device or printed.

• Text editors are used for file modification. In this, new file is created and content of file is modified. Programming language support includes the compilers, assemblers, interpreters for common programming language like C, C++, Java, Visual Basic.

• For program loading and execution, it is loaded into memory then program is executed by processor. Operating system provides different types loaders and linkers to complete execution operation.

• Debugging facility is provided by the operating system with the help of application program. It is used for checking errors.

• Communication between two devices are performed by creating temporary connection. Communication is in between process, users and other I/O devices. File transfer, remote login, electronic mail, browsing web, downloading multimedia data are the example of communication.

Design and Implementation

Design:

• Before designing the system, first define goals and specification of the system. Goals and specification will give the idea about the system and it also understands the requirement of the user. At the higher level, system design is dominated by the choice of hardware and system type.

• System type may be batch system, multiprogramming, multitasking, real time and distributed system. Hardware must support all these type of operating system.

• Once the goal and specification is clearly understood, then a requirement is related to user and system. These requirements are called user goals and system goals.

• User goals are as follows:

1. Simple to use and understand.

2. Provides reliability.

3. Provides security.

4. Fast speed.

• System goals are as follows:

1. Easy to design

2. Implementation and maintenance is also simple and easy

3. Free from error

4. Efficient to use.

• Operating system requirements is not fixed. It will change according to the user requirements. Single user operating system is required and multiple user operating systems is also required.

Implementation

• At first, operating systems were written in assembly, but now a days C/C++ is the language commonly used.

• Small blocks of assembly code are still needed, especially related to some low level I/O functions in device drivers, turning interrupt on and off and the Test and Set instruction for synchronization facilities.

• Using higher level language allow code to be written faster, is easier to read, and can be debugged easier. It also makes the OS much easier to port to different hardware platforms.