Operating System

Introductory Concepts: Historical evolution of operating systems, Operating system functions and characteristics, Classification: Real time systems, Time Sharing, Network Systems, Distributed systems, Batch Processing, Multitasking, Multiprogramming, Multiprocessing, System call, Basic Thread mechanics, Kernel Vs User level threads, Multithreading models

What is Operating System?

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a 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.

Generally, a **Computer System** consists of the following components:

- **Computer Users** are the users who use the overall computer system.
- **Application Softwares** are the softwares which users use directly to perform different activities. These softwares are simple and easy to use like Browsers, Word, Excel, different Editors, Games etc. These are usually written in high-level languages, such as Python, Java and C++.
- **System Softwares** are the softwares which are more complex in nature and they are more near to computer hardware. These software are usually written in low-level languages like assembly language and includes **Operating Systems** (Microsoft Windows, macOS, and Linux), Compiler, and Assembler etc.
- **Computer Hardware** includes Monitor, Keyboard, CPU, Disks, Memory, etc.

So now let's put it in simple words:

If we consider a Computer Hardware is body of the Computer System, then we can say an Operating System is its soul which brings it alive ie. operational. We can never use a Computer System if it does not have an Operating System installed on it.

Operating System - Examples

There are plenty of Operating Systems available in the market which include paid and unpaid (Open Source). Following are the examples of the few most popular Operating Systems:

- **Windows:** This is one of the most popular and commercial operating systems developed and marketed by Microsoft. It has different versions in the market like Windows 8, Windows 10 etc and most of them are paid.
- **Linux** This is a Unix based and the most loved operating system first released on September 17, 1991 by Linus Torvalds. Today, it has 30+ variants available like Fedora, OpenSUSE, CentOS, UBuntu etc. Most of them are available free of charges though you can have their enterprise versions by paying a nominal license fee.
- **MacOS** This is again a kind of Unix operating system developed and marketed by Apple Inc. since 2001.
- **iOS** This is a mobile operating system created and developed by Apple Inc. exclusively for its mobile devices like iPhone and iPad etc.
- **Android** This is a mobile Operating System based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

Some other old but popular Operating Systems include Solaris, VMS, OS/400, AIX, z/OS, etc.

Operating System - Functions

To brief, Following are some of important functions of an operating System which we will look in more detail in upcoming chapters:

- Process Management
- I/O Device Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- Security Management
- Command Interpreter System
- Control over system performance
- Job Accounting
- Error Detection and Correction
- Coordination between other software and users
- Many more other important tasks

Operating Systems - History

Operating systems have been evolving through the years. In the 1950s, computers were limited to running one program at a time like a calculator, but later in the following decades, computers began to include more and more software programs, sometimes called libraries, that formed the basis for today's operating systems.

The first Operating System was created by General Motors in 1956 to run a single IBM mainframe computer, its name was the IBM 704. IBM was the first computer manufacturer to develop operating systems and distribute them in its computers in the 1960s.

There are few facts about Operating System evaluation:

- Stanford Research Institute developed the oN-Line System (NLS) in the late 1960s, which was the first operating system that resembled the desktop operating system we use today.
- Microsoft bought QDOS (Quick and Dirty Operating System) in 1981 and branded it as Microsoft Operating System (MS-DOS). As of 1994, Microsoft had stopped supporting MS-DOS.
- Unix was developed in the mid-1960s by the Massachusetts Institute of Technology, AT&T Bell Labs, and General Electric as a joint effort. Initially it was named MULTICS, which stands for Multiplexed Operating and Computing System.
- FreeBSD is also a popular UNIX derivative, originating from the BSD project at Berkeley. All modern Macintosh computers run a modified version of FreeBSD (OS X).
- Windows 95 is a consumer-oriented graphical user interface-based operating system built on top of MS-DOS. It was released on August 24, 1995 by Microsoft as part of its Windows 9x family of operating systems.
- Solaris is a proprietary Unix operating system originally developed by Sun Microsystems in 1991. After the Sun acquisition by Oracle in 2010 it was renamed Oracle Solaris.

Why to Learn Operating System

If you are aspiring to become a Great Computer Programmer then it is highly recommended to understand how exactly an Operating System works inside out. This gives opportunity to understand how exactly data is saved in the disk, how different processes are created and scheduled to run by the CPU, how to interact with different I/O devices and ports.

There are various low level concepts which help a programmer to Design and Develop scalable softwares. Bottom line is without a good understanding of Operating System Concepts, it can't be assumed someone to be a good Computer **Application Software** developer, and even it is unimaginable imagine someone to become a **System Software** developer without knowing Operating System in-depth.

If you are a fresher and applying for a job in any standard company like Google, Microsoft, Amazon, IBM etc then it is very much possible that you will be asked questions related to Operating System concepts.

Target Audience

This tutorial has been prepared for the Computer Science Professionals and Students specially for BCA, MCA, B.Tech, M.Tech Engineering Students to help them understand the basic to advanced concepts related to an Operating System in general. Operating System is one of the core concepts in every University teaching Computer Science and this subject has a lot of weight from exams point of view.

Prerequisites

Before you start learning Operating System using this tutorial, we are making an assumption that you are already aware of Computer Fundaments like What is Computer Hardware, CPU, Primary Memory, Secondary Memory, Devices, Files etc. If you are not already aware of these concepts then it will be difficult to understand various concepts related to Operating System and so it is highly recommended to go through our Computer Fundamentals Tutorial before attempting to learn Operating System.

Functions of Operating System

- Processor management
- Act as a Resource Manager
- Memory Management
- File Management
- Security
- Device Management
- Input devices / Output devices
- Deadlock Prevention
- Time Management
- Coordinate with system software or hardware

Characteristics of Operating System

Here is a list of some of the most prominent characteristic features of Operating Systems −

- Memory Management − Keeps track of the primary memory, i.e. what part of it is in use by whom, what part is not in use, etc. and allocates the memory when a process or program requests it.
- Processor Management − Allocates the processor (CPU) to a process and deallocates the processor when it is no longer required.
- Device Management − Keeps track of all the devices. This is also called I/O controller that decides which process gets the device, when, and for how much time.
- File Management − Allocates and de-allocates the resources and decides who gets the resources.
- Security − Prevents unauthorized access to programs and data by means of passwords and other similar techniques.
- Job Accounting − Keeps track of time and resources used by various jobs and/or users.
- Control Over System Performance − Records delays between the request for a service and from the system.
- Interaction with the Operators − Interaction may take place via the console of the computer in the form of instructions. The Operating System acknowledges the same, does the corresponding action, and informs the operation by a display screen.
- Error-detecting Aids − Production of dumps, traces, error messages, and other debugging and error-detecting methods.
- Coordination Between Other Software and Users − Coordination and assignment of compilers, interpreters, assemblers, and other software to the various users of the computer systems.

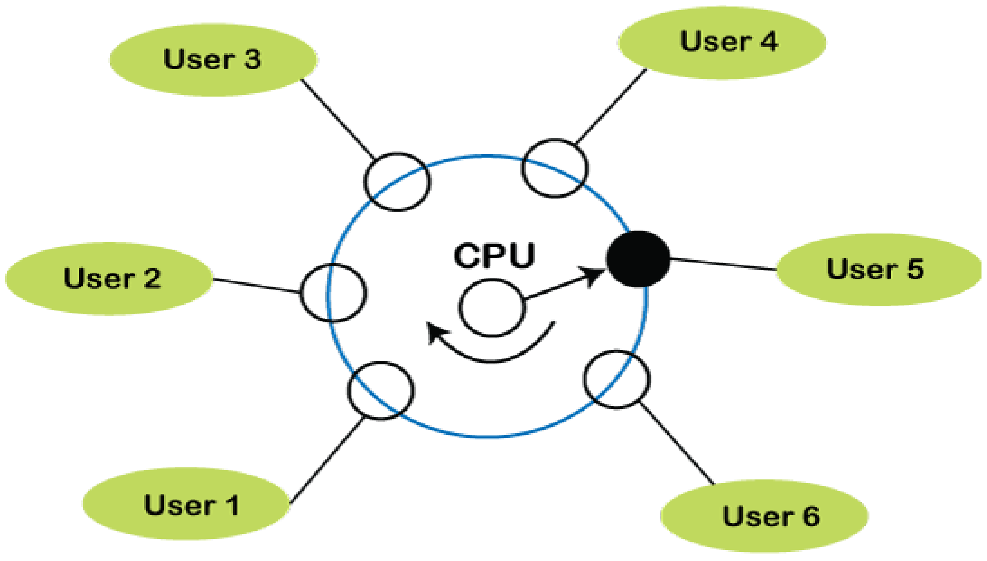
Types of Operating System

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
9. Multitasking

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.

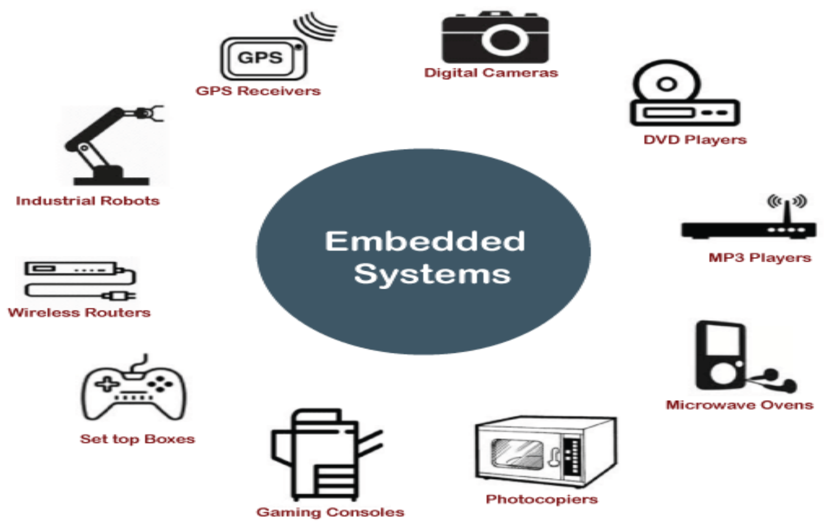
Time-Sharing Operating System



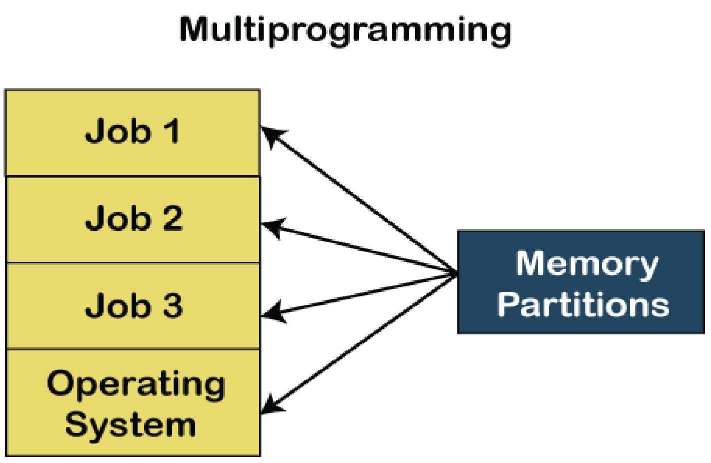
It is the type of operating system that allows us to connect many people located at different locations to share and use a specific system at a single time. The time-sharing operating system is the logical extension of the multiprogramming through which users can run multiple tasks concurrently. Furthermore, 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.

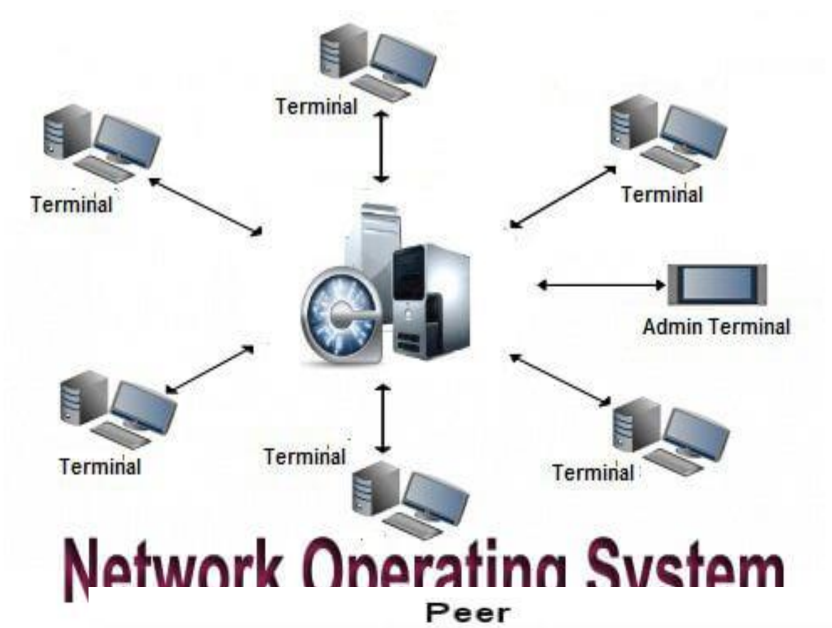


Multiprogramming Operating System



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.

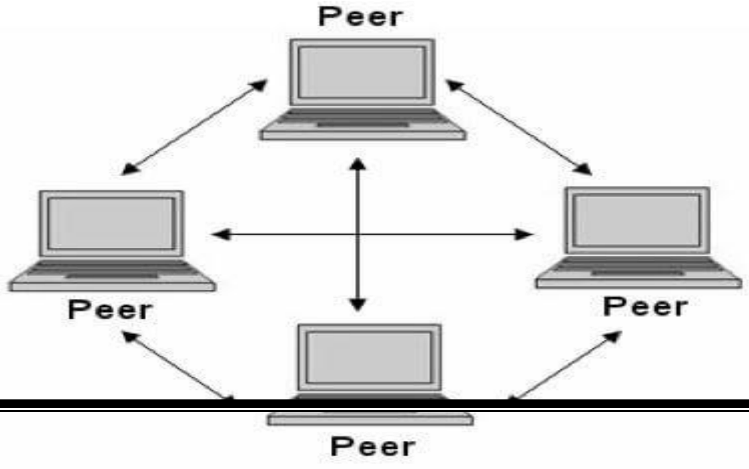
Network Operating System



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.

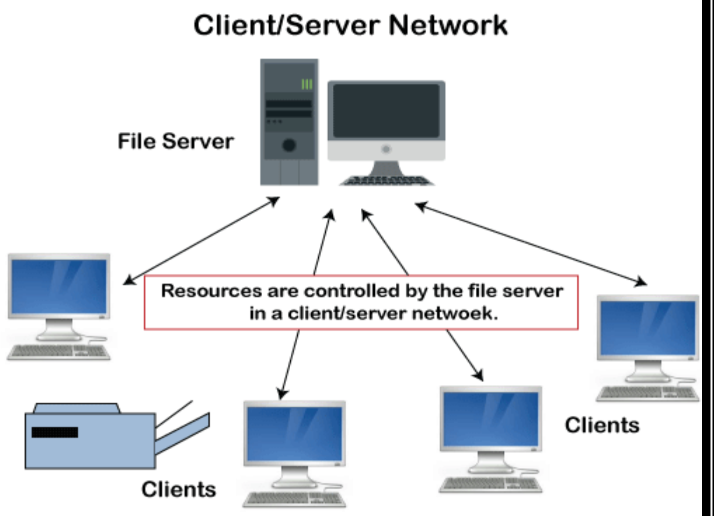
Types of network operating system

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



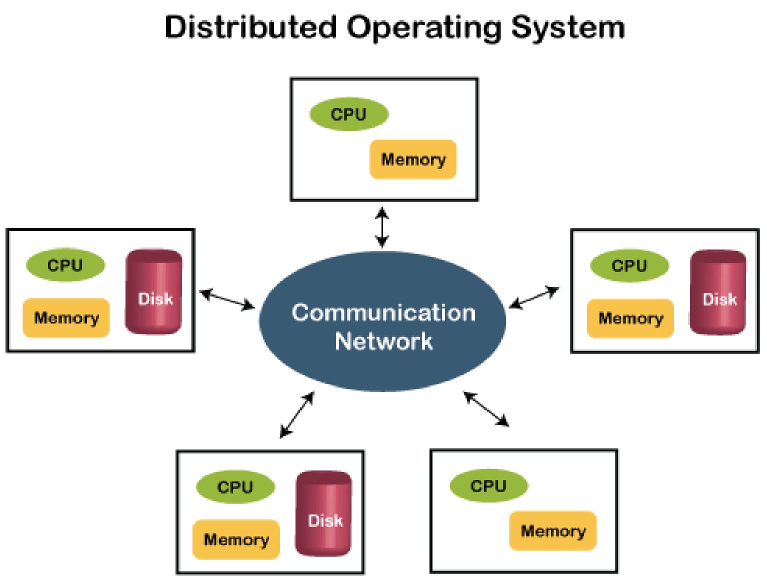
more computer machines using a LAN.

- **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.


Client/Server Network
File Server
Resources are controlled by the file server in a client/server netwoek.
Clients
Clients

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.

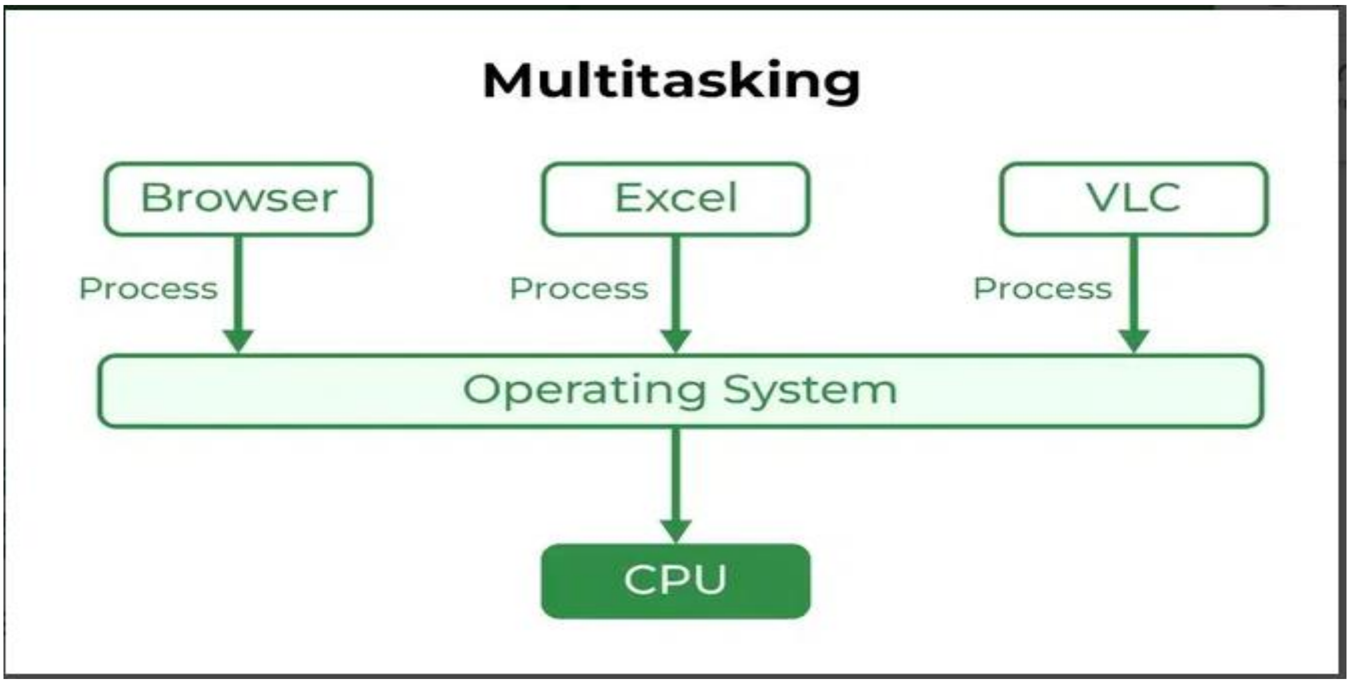
Types of the real-time operating system:

- **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.

9Multi-Tasking Operating System

Multitasking Operating System is simply a multiprogramming Operating System with having facility of a Round-Robin Scheduling Algorithm. It can run multiple programs simultaneously.
There are two types of Multi-Tasking Systems which are listed below.

- Preemptive Multi-Tasking
- Cooperative Multi-Tasking



Multitasking

Advantages of Multi-Tasking Operating System

- Multiple Programs can be executed simultaneously in Multi-Tasking Operating System.
- It comes with proper memory management.

Disadvantages of Multi-Tasking Operating System

- The system gets heated in case of heavy programs multiple times.

Generations of Operating System

The First Generation (1940 to early 1950s)

When the first electronic computer was developed in 1940, it was created without any operating system. In early times, users have full access to the computer machine and write a program for each task in absolute machine language. The programmer can perform and solve only simple mathematical calculations during the computer generation, and this calculation does not require an operating system.

The Second Generation (1955 - 1965)

The first operating system (OS) was created in the early 1950s and was known as **GMOS. General Motors** has developed OS for the **IBM** computer. The second-generation operating system was based on a 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. At each completion of jobs (either normally or abnormally), control transfer to the operating system that is cleaned after completing one job and then continues to read and initiates the next job in a punch card. After that, new machines were called mainframes, which were very big and used by professional operators.

The Third Generation (1965 - 1980)

During the late 1960s, operating system designers were very capable of developing a new operating system that could simultaneously perform multiple tasks in a single computer program called multiprogramming. The introduction of **multiprogramming** plays a very important role in developing operating systems that allow a CPU to be busy every time by performing different tasks on a computer at the same time. During the third generation, there was a new development of minicomputer's phenomenal growth starting in 1961 with the DEC PDP-1. These PDP's leads to the creation of personal computers in the fourth generation.

The Fourth Generation (1980 - Present Day)

The fourth generation of operating systems is related to the development of the personal computer. However, the personal computer is very similar to the minicomputers that were developed in the third generation. The cost of a personal computer was very high at that time; there were small fractions of minicomputers costs. A major factor related to creating personal computers was the birth of Microsoft and the Windows operating system. Microsoft created the first **window** operating system in 1975. After introducing the Microsoft Windows OS, Bill Gates and Paul Allen had the vision to take personal computers to the next level. Therefore, they introduced the **MS-DOS** in 1981; however, it was very difficult for the person to understand its cryptic commands. Today, Windows has become the most popular and most commonly used operating system technology. And then, Windows released various operating systems such as Windows 95, Windows 98, Windows XP and the latest operating system, Windows 7. Currently, most Windows users use the Windows 10 operating system. Besides the Windows operating system, Apple is another popular operating system built in the 1980s, and this operating system was developed by Steve Jobs, a co-founder of Apple. They named the operating system Macintosh OS or Mac OS.

Advantages of Operating System

- It is helpful to monitor and regulate resources.
- It can easily operate since it has a basic graphical user interface to communicate with your device.
- It is used to create interaction between the users and the computer application or hardware.
- The performance of the computer system is based on the CPU.
- The response time and throughput time of any process or program are fast.
- It can share different resources like fax, printer, etc.
- It also offers a forum for various types of applications like system and web application.

Disadvantage of the Operating System

- It allows only a few tasks that can run at the same time.
- It any error occurred in the operating system; the stored data can be destroyed.
- It is a very difficult task or works for the OS to provide entire security from the viruses because any threat or virus can occur at any time in a system.
- An unknown user can easily use any system without the permission of the original user.
- The cost of operating system costs is very high.

What is a System Call?

A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.

The **Application Program Interface (API)** connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.

How are system calls made?

When a computer software needs to access the operating system's kernel, it makes a system call. The system call uses an API to expose the operating system's services to user programs. It is the only method to access the kernel system. All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.

Below are some examples of how a system call varies from a user function.

1. A system call function may create and use kernel processes to execute the asynchronous processing.
2. A system call has greater authority than a standard subroutine. A system call with kernel-mode privilege executes in the kernel protection domain.
3. System calls are not permitted to use shared libraries or any symbols that are not present in the kernel protection domain.
4. The code and data for system calls are stored in global kernel memory.

Why do you need system calls in Operating System?

There are various situations where you must require system calls in the operating system. Following of the situations are as follows:

1. It is must require when a file system wants to create or delete a file.
2. Network connections require the system calls to sending and receiving data packets.
3. If you want to read or write a file, you need to system calls.
4. If you want to access hardware devices, including a printer, scanner, you need a system call.
5. System calls are used to create and manage new processes.

How System Calls Work

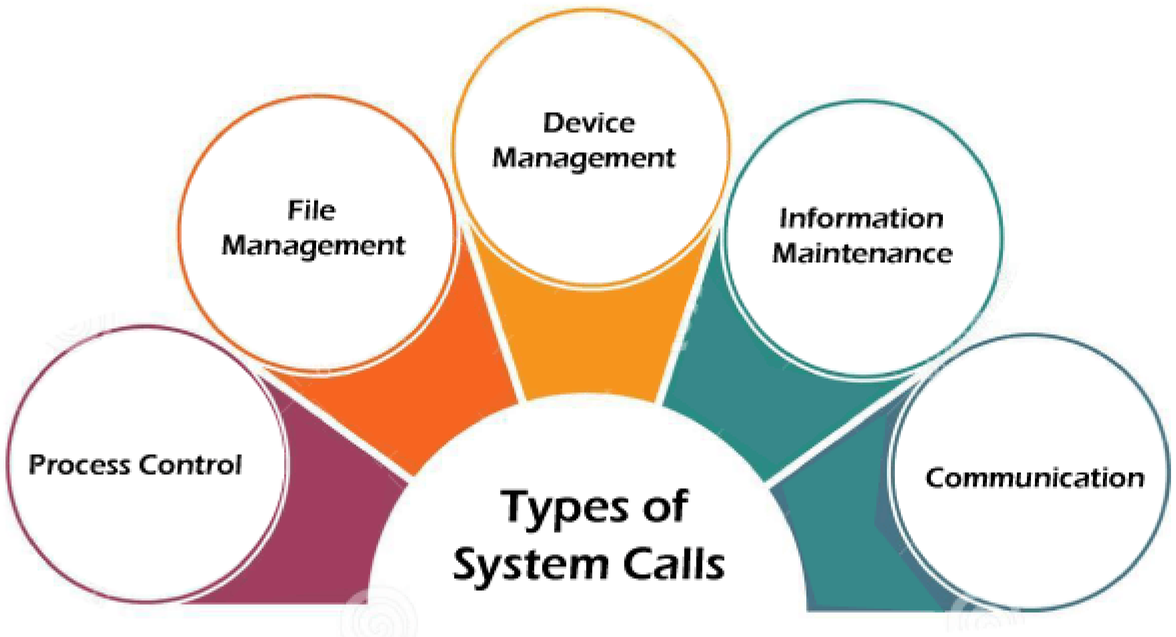
The Applications run in an area of memory known as user space. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel. It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.

If the request is permitted, the kernel performs the requested action, like creating or deleting a file. As input, the application receives the kernel's output. The application resumes the procedure after the input is received. When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.

A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

Types of System Calls

There are commonly five types of system calls. These are as follows:



1. **Process Control**
2. **File Management**
3. **Device Management**
4. **Information Maintenance**
5. **Communication**

Now, you will learn about all the different types of system calls one-by-one.

Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

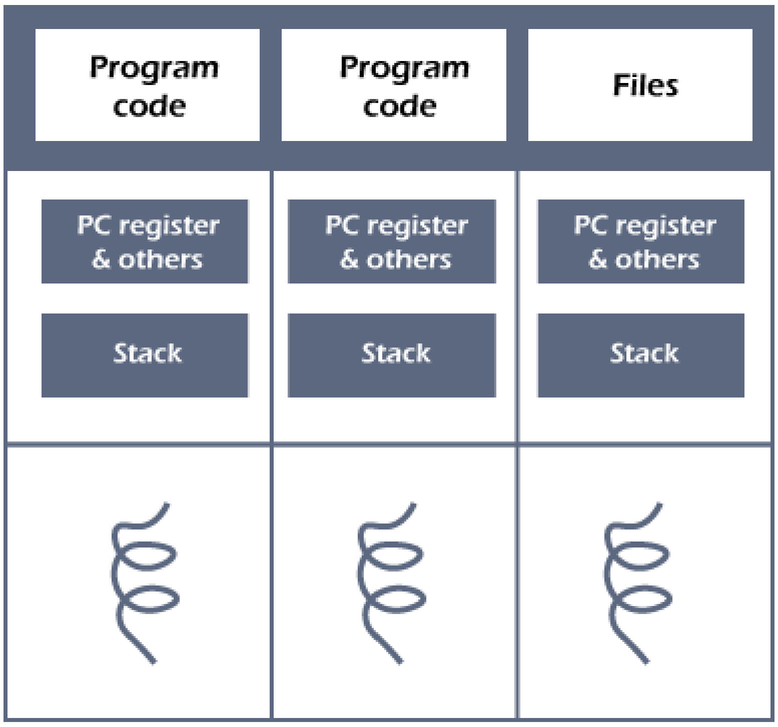
Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Threads in Operating System (OS)

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



Three threads of same process

The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Need of Thread:

- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use Inter- Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User-level thread

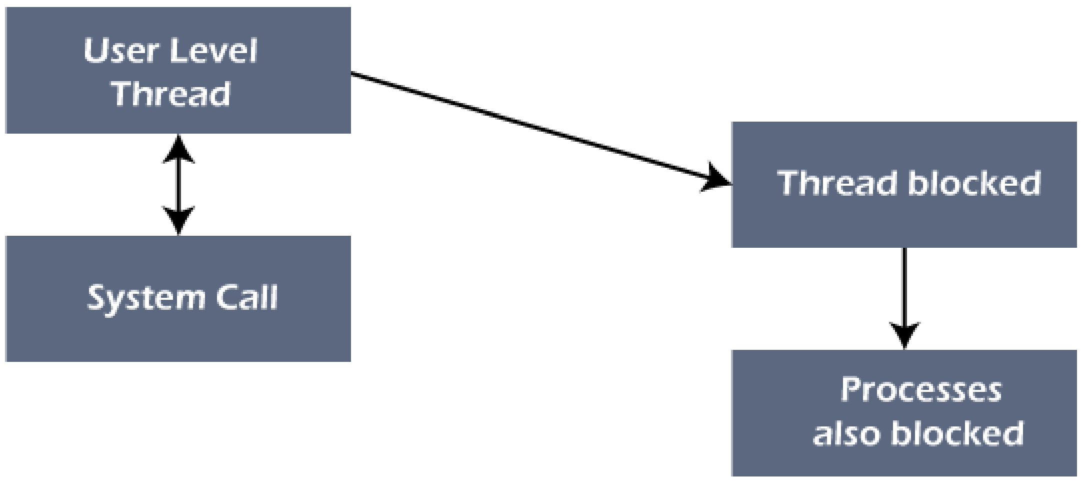
The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes?examples: Java thread, POSIX threads, etc.

Advantages of User-level threads

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

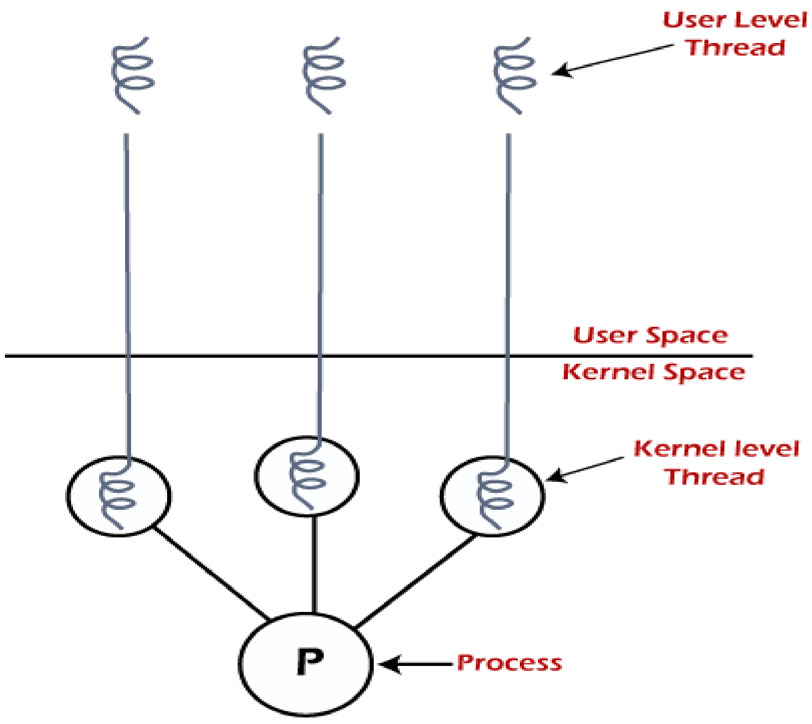
Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.



Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

Benefits of Threads

- **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.

- **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
- **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.