UNIT-I

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles

Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - **polling**
  - **vectored** interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline

I/O Structure

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the operating system to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- 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

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage

Storage-Device Hierarchy

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

Computer-System Architecture

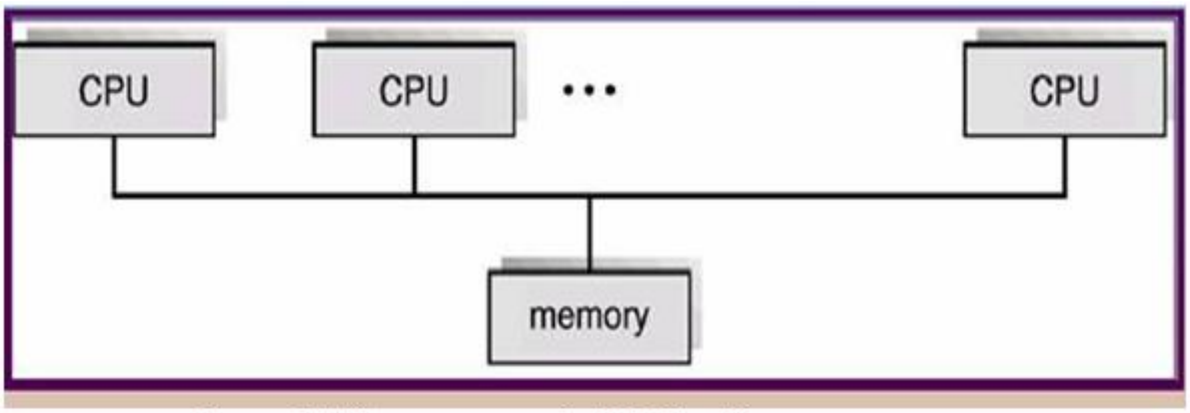


Figure 1.6 illustrates a typical SMP architecture.

A computer system may be organized in a number of different ways, which we can categorize roughly according to the number of general-purpose processors used.

· **1.3.1 Single-Processor Systems**

Most systems vise a single processor. a single-processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes. Almost all systems have other special-purpose processors as well. They may come in the form of device-specific processors, such as disk, keyboard, and graphics controllers; or, on mainframes, they may come in the form of more general-purpose processors, such as I/O processors that move data rapidly among the components of the system. All of these special-purpose processors run a limited instruction set and do not run user processes. Sometimes they are managed by the operating system, in that the operating system sends them information about their next task and monitors their status.

· **1.3.2 Multiprocessor Systems**

Multiprocessor systems (also known as **parallel systems** or **tightly coupled systems)** are growing in importance. Such systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

Multiprocessor systems have three main advantages:

1. Increased throughput. By increasing the number of processors, we expect to get more work done in less time. The speed-up ratio with *N* processors is not *N*, however; rather, it is less than N. When multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly. This overhead, plus contention for shared resources, lowers the expected gain from additional processors. Similarly, *N* programmers working closely together do not produce *N* times the amount of work a single programmer would produce.

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

3. **Increased reliability.** If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fails, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether.

Graceful degradation is the ability to continue providing service proportional to the level of surviving hardware.
Fault tolerant some systems go beyond graceful degradation.

The multiple-processor systems in use today are of two types.

· **Asymmetric multiprocessing,** which each processor is assigned a specific task.
· **Symmetric multiprocessing (SMP),** The most common systems use in which each processor performs all tasks within the operating system.

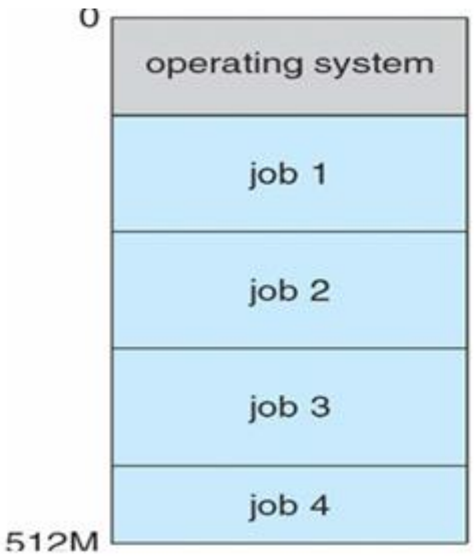


Figure 1.7 Memory layout for a multiprogramming system.

· 1.3.3 Clustered Systems

Clustered system is an another type of multiple-CPU system. Clustered systems differ from multiprocessor systems, however, in that they are composed of two or more individual systems coupled together. The definition of the term *clustered* is not concrete; many commercial packages wrestle with what a clustered system is and why one form is better than another. The generally accepted definition is that clustered computers share storage and are closely linked via a **local-area network (LAN)** or a faster interconnect such as InfiniBand.

Clustering is usually used to provide **high-availability** service; that is, service will continue even if one or more systems in the cluster fail. Clustering can be structured asymmetrically or symmetrically. Cluster technology is changing rapidly. Some cluster products support dozens of systems in a cluster, as well as clustered nodes that are separated by miles. Many of these improvements are made possible by **storage-area networks (SANs),** as described in Section 12.3.3, which allow many systems to attach to a pool of storage.

· 1.4Operating-System Structure

An operating system provides the environment within which programs are executed. One of the most important aspects of operating systems is the ability to multiprogram. A single user cannot, in general, keep either the CPU or the I/O devices busy at all times.

Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute. The operating system keeps several jobs in memory simultaneously (Figure 1.7).

Time sharing (or **multitasking)** is a logical extension of multiprogramming. In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

Time sharing requires an **interactive** (or **hands-on) computer system,** which provides direct communication between the user and the system.

A time-shared operating system allows many users to share the computer simultaneously. A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

1.5 Operating-System Operations

Interrupt driven is modern operating systems.

Trap (or an **exception)** is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed. The interrupt-driven nature of an operating system defines that system's general structure. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt.

Since the operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program that was running.

· 1.5.1 Dual-Mode Operation

Two separate **modes** of operation: **user mode** and **kernel mode** (also called **supervisor mode, system mode, or privileged mode).** A bit, called the **mode bit,** is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).

The dual mode of operation provides us with the means for protecting the operating system from errant users—and errant users from one another.

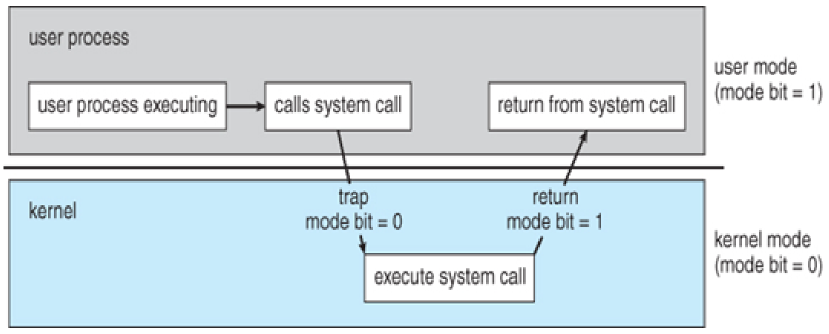


Figure 1.8 Transition from user to kernel mode.

We accomplish this protection by designating some of the machine instructions that may cause harm as **privileged instructions.** The hardware allows privileged instructions to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

· **1.5.2 Timer**

A **timer** can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second). **Variable timer** is generally implemented by a fixed-rate clock and a counter. The operating system sets the counter. Every time the clock ticks, the counter is decremented. We can use the timer to prevent a user program from running too long. A simple technique is to initialize a counter with the amount of time that a program is allowed to run.

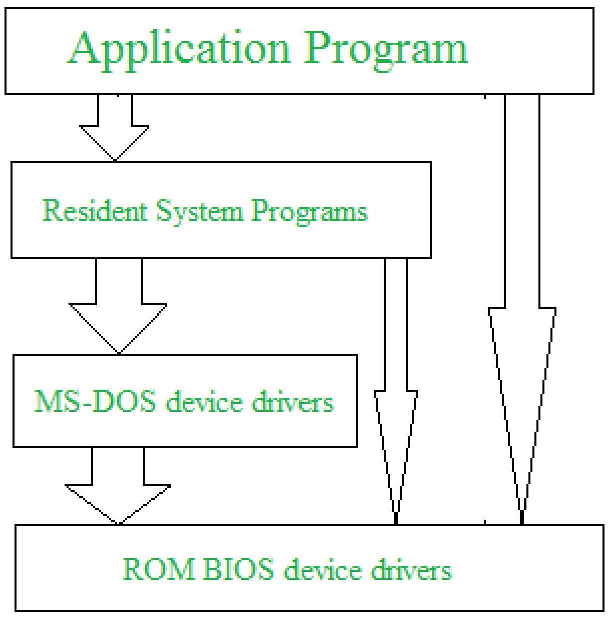
Structures of Operating Systems

Operating system can be implemented with the help of various structures. The structure of the OS depends mainly on how the various common components of the operating system are interconnected and melded into the kernel. Depending on this we have following structures of the operating system:

Simple structure:

Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O

routines. These types of operating system cause the entire system to crash if one of the user programs fails.
Diagram of the structure of MS-DOS is shown below.



Advantages of Simple structure:

- It delivers better application performance because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.

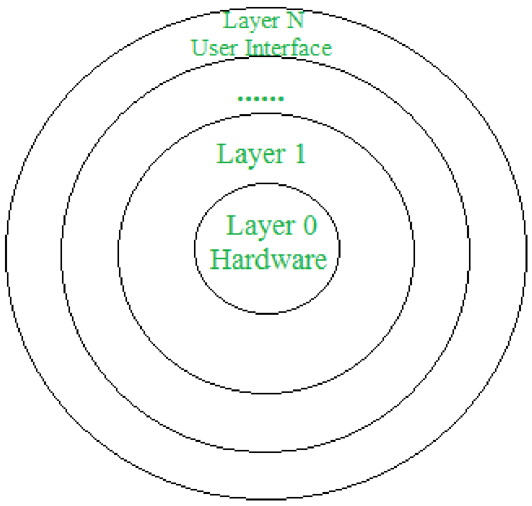
Disadvantages of Simple structure:

- The structure is very complicated as no clear boundaries exists between modules.
- It does not enforce data hiding in the operating system.

Layered structure:

An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.



Advantages of Layered structure:

- Layering makes it easier to enhance the operating system as implementation of a layer can be changed easily without affecting the other layers.
- It is very easy to perform debugging and system verification.

Disadvantages of Layered structure:

- In this structure the application performance is degraded as compared to simple structure.
- It requires careful planning for designing the layers as higher layers use the functionalities of only the lower layers.

Micro-kernel:

This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This result in a smaller kernel called the micro-kernel.
Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

Advantages of Micro-kernel structure:

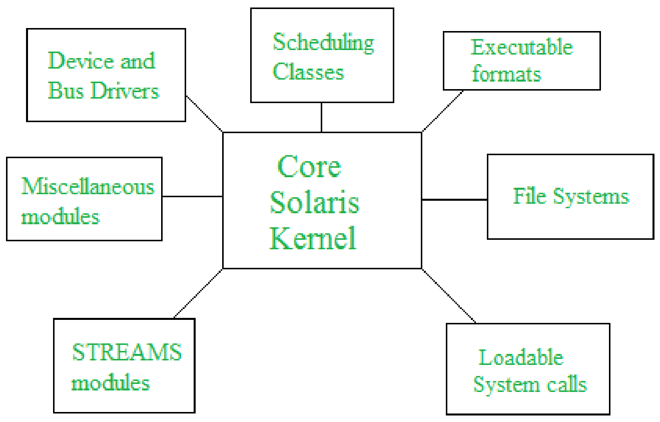
- It makes the operating system portable to various platforms.
- As microkernels are small so these can be tested effectively.

Disadvantages of Micro-kernel structure:

- Increased level of inter module communication degrades system performance.

Modular structure or approach:

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module. For example Solaris OS is organized as shown in the figure.



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

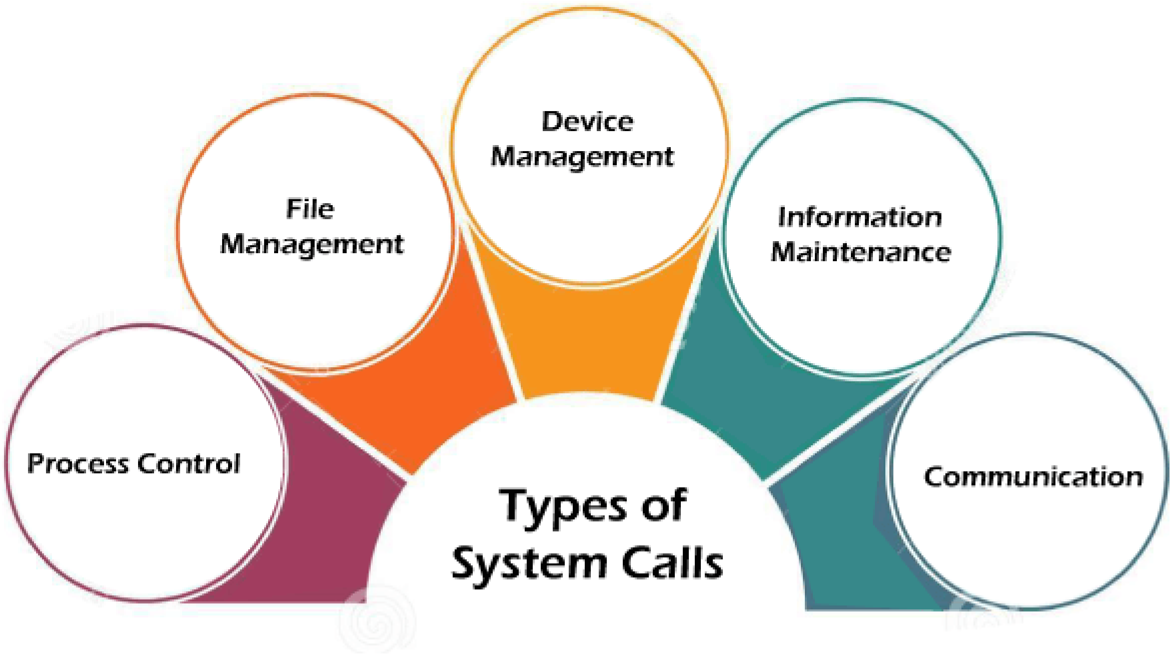
The Applications run in an area of memory known as user space. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel. It achieves this using an interrupt request, which pauses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

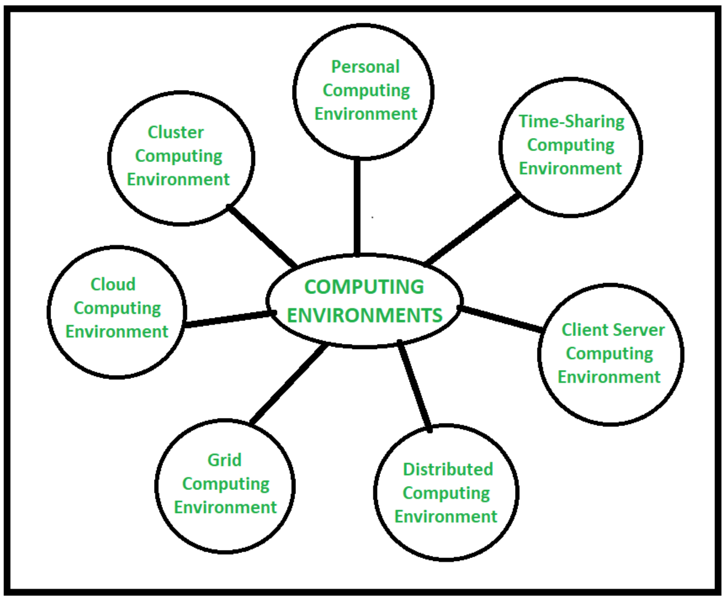
Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Computing Environments

Computing Environments : When a problem is solved by the computer, during that computer uses many devices, arranged in different ways and which work together to solve problems. This constitutes a computing environment where various number of computer devices arranged in different ways to solve different types of problems in different ways. In different computing environments computer devices are arranged in different ways and they exchange information in between them to process and solve problem. One computing environment consists of many computers other computational devices, software and networks that to support processing and sharing information and solving task. Based on the organization of different computer devices and communication processes there exists multiple types of **computing environments**.

Now lets know about different types of computing environments.

Types of Computing Environments : There are the various types of computing environments. They are :


Personal Computing Environment
Cluster Computing Environment
Time-Sharing Computing Environment
Cloud Computing Environment
COMPUTING ENVIRONMENTS
Client Server Computing Environment
Grid Computing Environment
Distributed Computing Environment

Computing Environments Types

1. **Personal Computing Environment :** In personal computing environment there is a stand-alone machine. Complete program resides on computer and executed there. Different stand-alone machines that constitute a personal computing environment are laptops, mobiles, printers, computer systems, scanners etc. That we use at our homes and offices.
2. **Time-Sharing Computing Environment :** In Time Sharing Computing Environment multiple users share system simultaneously. Different users (different processes) are allotted different time slice and processor switches rapidly among users according to it. For example, student listening to music while coding something in an IDE. Windows 95 and later versions, Unix, IOS, Linux operating systems are the examples of this time sharing computing environment.

3. **Client Server Computing Environment :** In client server computing environment two machines are involved i.e., client machine and server machine, sometime same machine also serve as client and server. In this computing environment client requests resource/service and server provides that respective resource/service. A server can provide service to multiple clients at a time and here mainly communication happens through computer network.
4. **Distributed Computing Environment :** In a distributed computing environment multiple nodes are connected together using network but physically they are separated. A single task is performed by different functional units of different nodes of distributed unit. Here different programs of an application run simultaneously on different nodes, and communication happens in between different nodes of this system over network to solve task.
5. **Grid Computing Environment :** In grid computing environment, multiple computers from different locations works on single problem. In this system set of computer nodes running in cluster jointly perform a given task by applying resources of multiple computers/nodes. It is network of computing environment where several scattered resources provide running environment for single task.
6. **Cloud Computing Environment :** In cloud computing environment on demand availability of computer system resources like processing and storage are availed. Here computing is not done in individual technology or computer rather it is computed in cloud of computers where all required resources are provided by cloud vendor. This environment primarily comprised of three services
7. **Cluster Computing Environment :** In cluster computing environment cluster performs task where cluster is a set of loosely or tightly connected computers that work together. It is viewed as single system and performs task parallelly that’s why also it is similar to parallel computing environment. Cluster aware applications are especially used in cluster computing environment.

Open-Source Operating Systems.

The term **"open source"** refers to computer software or applications where the owners or copyright holders enable the users or third parties to use, see, and edit the product's source code. The source code of an open-source OS is publicly visible and editable. The usually operating systems such as Apple's iOS, Microsoft's Windows, and Apple's Mac OS are closed operating systems. Open-Source Software is licensed in such a way that it is permissible to produce as many copies as you want and to use them wherever you like. The open-source operating system allows the use of code that is freely distributed and available to anyone and for commercial purposes. Being an open-source application or program, the program source code of an open-source OS is available. The user may modify or change those codes and develop new applications according to the user requirement. Some basic examples of the open-source operating systems are **Linux, Open Solaris, Free RTOS, Open BDS, Free BSD, Minix,** etc.

How does Open-Source Operating System work?

It works similarly to a closed operating system, except that the user may modify the source code of the program or application. There may be a difference in function even if there is no difference in performance.

For instance, the information is packed and stored in a proprietary (closed) operating system. In open-source, the same thing happens. However, because the source code is visible to you, you may better understand the process and change how data is processed.

While the former operating system is secure and hassle-free, and the latter requires some technical knowledge, you may customize these and increase performance. There is no specific way or framework for working on the open-source OS, but it may be customized on the user requirements.

Best Open-Source Operating System

Most of the open-source operating systems are Linux based. Some of the best open-source operating systems are as follows:

1. Linux Kernel

Linux kernel was developed by Linus Torvalds. It offers the essential functions required for an operating system, such as data cancellation, memory processing, and interactions with computer hardware. It is open-source software, and many developers researched the source code and produced a plethora of helpful plug-ins and operating systems to meet their requirements.

2. Linux Lite

Linux Lite is another free and open-source operating system that can run on lower-end hardware. It is a lightweight operating system designed to help users who are unfamiliar with Linux-based operating systems. The operating system includes all of the required programs, capabilities, tools, and desktops. It has a minimal interface and is entirely based on the Ubuntu system. In the last five years, the operating system has been stable and has received regular updates. It is efficiently functional soon after installation. After installation, users are not required to install any further drivers. If you want a lightweight open-source operating system on your PC, go with Linux Lite.

3. Linux mint

Linux Mint is a powerful Linux-based operating system that exudes modernity and power. It is simple to use and includes complete multimedia capabilities, making it a user-friendly open-source operating system. It is an Ubuntu-based distribution that is popular among both beginners and experts. It is built on the Debian platform and includes one of the most powerful software managers. It is more stable and has better visual aesthetics than Ubuntu.

4. Fedora

Fedora is another popular Linux-based operating system, and it is widely considered the best open-source operating system after Ubuntu. It is an RPM-based general-purpose operating system that is supported by Red Hat and built by the Fedora Project community. Its purpose is to develop and share cutting-edge open-source technology for free. As a result, Fedora developers prefer to make upstream improvements rather than create fixes specifically for Fedora. Fedora developers' updates are available to all Linux distributions.

It has a GNOME-based desktop that may be customized. Fedora comes with a customizable GNOME-based desktop. Its Fedora Spins feature allows you to customize and run several user interfaces and desktop environments.

5. React OS

ReactOS is another free and open-source operating system that has nearly 1 million downloads in over **100** countries. This community-based OS may run Windows apps, making it an excellent alternative to the Windows operating system. Although ReactOS is still growing, users, who love highly customizable operating systems, can select ReactOS. However, the operating system is developer-focused.

6. Solus

Solus is a free and open-source operating system for your desktop computer. It's a new operating system from the Linux family, released in **2012**. More than **6000** registered users are currently using the software. VLC, XChat, Transmission, Thunderbird, OpenShot Video Editor, Firefox, Budgie desktop environment, and LibreOffice Suite are all included with Solus. The most recent version of Solus, **Solus 3**, was released in **August 2017**.

7. Chrome OS

Chrome OS is a partly open-source operating system with various attractive features. It's a part of the Chromium and Linux families, with features including better security, compatibility for supported Android and Chrome apps, Aura windows manager, Google cloud print, integrated media player, virtual desktop access, and cloud-based management. The only issue with the operating system is that it only supports Nexus devices or its hardware. As a result, if you're a Google fan, you'll love Chrome OS on a Chromebook.

Advantages and Disadvantages of Open-Source Operating System

Various advantages and disadvantages of the open-source operating system are as follows:

Advantages

1. Reliable and efficient

The open-source operating systems are most reliable and efficient. Thousands of eyes monitor these because the source code is public. As a result, if there are any bugs or errors, they are fixed by the best developers worldwide.

2. Cost-efficient

Most of the open-source operating systems are free. And some of them are far less expensive than commercially closed products.

3. Flexibility

The great advantage is you may customize it as per your requirement. And there is creative freedom.

Disadvantages

1. Complicated

It is not as user-friendly as the ones that are closed. To use this software, you must have a basic understanding of technology.

2, Security risk

Despite the defects having been detected, there is a risk of assaults because the attackers have access to the source code.

3. No support

If you run across an issue, there is no customer support available to assist you.