CHAPTER

1. Introduction to Operating System

[** This topic is not part of the GTU syllabus. But this chapter has included it to understand the basic concepts of operating systems thoroughly.]

1.1 INTRODUCTION

What is a Computer...?

It is a device consisting of a processor, memory, keyboard, mouse, disk, monitor and many other peripherals. In short, somewhat complex system...!!! But this is only the hardware and not enough for a system to work. Besides hardware, a computer system also contains software parts. The most important software is Operating System. It is the soul of the computer system, without which the entire system is just like a Dead (human) body.

This chapter provides an overview of the Operating System, formerly known as OS. The chapter begins with a description of the need for an Operating System. Then, it goes through the history of Operating Systems very briefly. This follows the description of various types of Operating Systems. After this, the services of the Operating System are described from two different viewpoints: User and System. Two well-known Operating Systems – Linux and Windows XP – are described next as a case study. In the end, some supplementary topics are given, which are not parts of the syllabus but necessary to get a proper understanding and thorough knowledge of the Operating System. These topics include the architecture of the operating systems, processor execution modes and system calls.

1.2 NEED FOR OPERATING SYSTEM

This section defines the Operating System and specifies the primary goals of Operating Systems. After this, Operating System is described as an essential component of a computer system. In the end, the need for Operating System is described from two different viewpoints.

1.2.1 Definition and Goal

➢ **Definition :**

Operating System is a program/software that –

- Manages the computer hardware, and
- Provides a simple interface to the hardware for user programs.

➢ **Goal :**

There are two main goals of the Operating System.

- Convenience for users; and
- Efficient operation of the computer system.

1.2.2 Operating System: An Essential Component of Computer System

The computer system can be divided into four main components: Hardware, Operating System, Application Programs and Users. Figure 1.1 shows the positions of each component

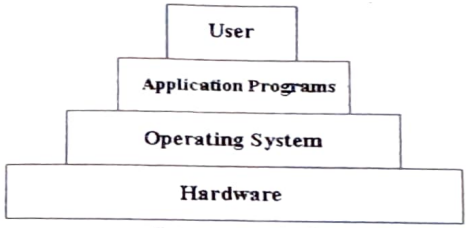

Figure 1.1 : Computer System

Each of these components is described below:

➢ **Hardware :**

The hardware comes at the lowest level. It contains various kinds of physical devices. Such devices include processors, memory, keyboard, mouse, monitor, bus, etc.

➢ **Operating System :**

The next level is for the Operating System. It manages all the underlying hardware. It also masks the complex details of hardware from the user. Thus, provides a simple interface between application programs and hardware.

➢ **Application Programs :**

Application programs perform particular tasks. They use different functionalities provided by the Operating System to perform their tasks. Examples of application programs include airline reservation systems, banking systems, web-browser, and many others.

➢ Users :

Users are at the top. Users interact with the system by using application programs to perform particular tasks.

1.2.3 Operating System: From the User and System view

➢ **Operating System: from the user view**

- This is a top-down view.
- From a user point of view, OS can be considered an Extended Machine.
- Here, OS represents the computer system as a machine that is easier to program than the underlying hardware.
- OS hides all the details of the hardware. It acts as a layer between the user and the hardware and provides a simple interface.
- The programmers (users) do not need to be concerned about the complexities of the hardware. They use the functionalities provided by the OS, such as system calls, to complete their work. (System calls are explained in section 1.9.)
- Here, the primary goal of OS is *user convenience*.

- **Operating System: from a system view**
  - This is a bottom-up view.
  - From a system point of view, OS can be considered a Resource Manager.
  - Here, OS manages all the resources such as CPU, memory, and I/O devices.
  - Resources are shared in one of two ways:
    i) By multiplexing them in Time: Each user takes a turn to use the resource. For example, CPU.
    ii) By multiplexing them in Space: Each user gets part of the resource. For example, Memory.
  - Here, the primary goal of OS is *efficiency*.

1.3 TYPES OF OPERATING SYSTEMS

Starting from the beginning, Operating Systems have evolved a lot. In this section, the details of different types of operating systems are explained. Various important Operating Systems are listed below:

- Batch Operating System
- Multiprogramming Operating System
- Time Sharing Operating System
- Real-Time Operating System.
- Distributed Operating System
- Multithreading Operating System
- Multi-user Operating System
- Multiprocessing Operating System
- Network Operating System

These Operating Systems are described in the following subsections.

1.3.1 Batch Operating System

In earlier days, computers were large machines. The common input devices were card readers and tape drives. The common output devices were line printers, punch cards and tape drives.

The entire system worked in the following manner:

- The Operating System was very simple and always resident in the main memory.
- Programmers would prepare a job and submit it to the operator. The job consisted of a program, data and some control information.
- An operator would sort them in batches with similar requirements and run them batch-wise as the computer became available.
- At some later time (may be after some hours or even some days), output appeared. The output consisted of the result of the program and error information. Programmers needed to wait during this time and then collect output from the operator.

Here, memory is divided into two parts, as given in Figure 1.2. It is shared between operating system and the job. At a time, one job is selected from the batch of jobs and loaded in memory for execution. Once its execution completes, another job is selected and loaded into memory for execution. This procedure continues until all the jobs in a batch get executed.

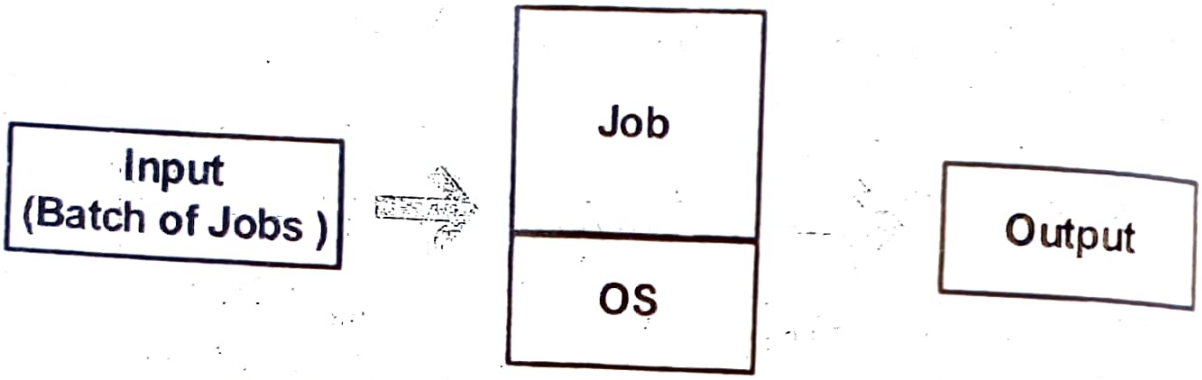

Figure 1.2 : Memory structure in Batch Operating System

The introduction of disk technology made it possible to access several jobs directly. And job scheduling came into the picture.

➢ **Disadvantages:**

- Low throughput: because the CPU remains idle when I/O is going on.
- Programmers do not have direct interaction with their jobs.
- Debugging is possible only offline after the output appears.
- Operations were too time-consuming.

1.3.2 Multiprogramming Operating System

A multiprogramming Operating System provides the ability to run more than one program concurrently.

In contrast to Batch Operating System, more than one program (or job) can simultaneously be loaded into the main memory. These programs can be executed concurrently. Memory is shared between OS and such kinds of programs, as shown in Figure 1.3.

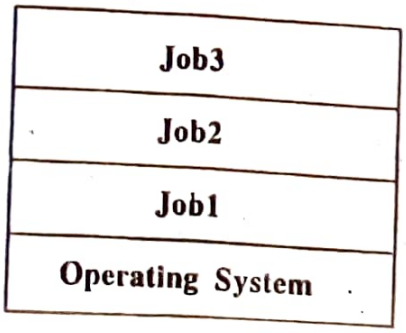

Figure 1.3 : Memory in Multiprogramming Operating System

➢ **Working :**

- A simple multiprogramming Operating System works in a non-preemptive manner.
- A program is allowed to execute until it voluntarily gives up the CPU.
- A program voluntarily gives up the CPU when it waits for some event, such as an I/O operation, or when it terminates.
- Once a CPU becomes free, it can be allocated to another program, as shown in Figure 1.4.

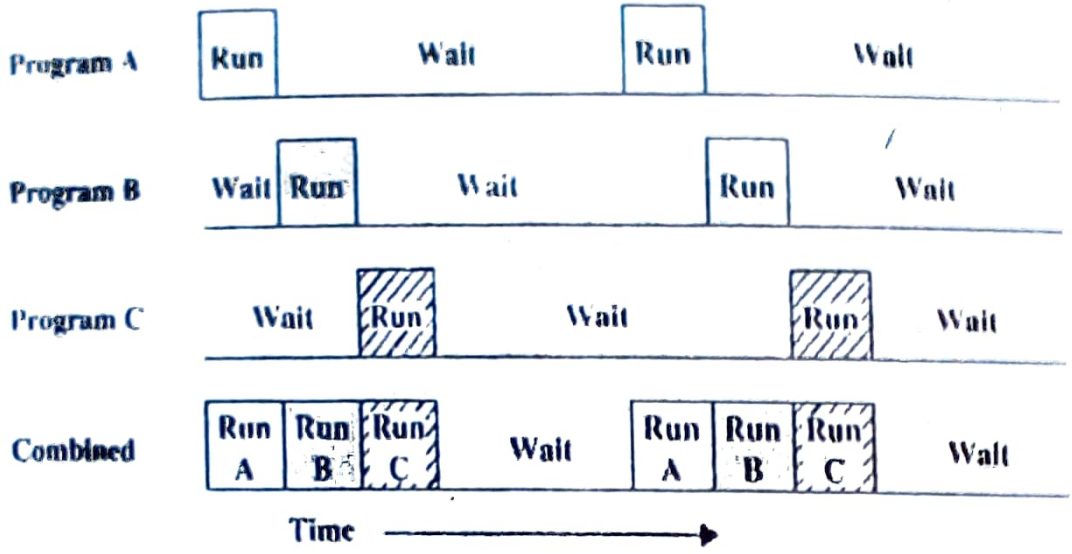

Figure 1.4 : Multiprogramming with three programs

➢ **Objective :**

- The primary objective of the Multiprogramming Operating System is to maximize CPU usage.

➢ **Advantages :**

- Multiprogramming significantly improves **system throughput** and **resource utilization**. This is because various resources, such as CPU, can be utilized as much as possible among various simultaneously executing programs.

1.3.3 Time Sharing Operating System

Time Sharing Operating System is a logical extension of the Multiprogramming Operating System. Here, the CPU is multiplexed by time among several programs (or jobs) that are kept in the main memory and on disk.

Pseudo-parallelism is achieved by providing rapid CPU scheduling among programs. Thus, users can have a good terminal response. It also provides direct interaction between the user and the system.

➢ **Working :**

- A Time Sharing Operating System works in a preemptive manner.
- Here, a program is allowed to execute only for some maximum time duration. After this time duration, a CPU is forcibly taken away from that program and allocated to another program.

- Here, smaller programs need not wait for another large program to finish its execution. So, it minimizes the response time for the user. And so they are suitable for interactive programs where users can directly interact with the program.
- This type of system is also referred to as a **multitasking** system. Users can execute more than one program and interact with them simultaneously. This means a user can perform more than one task simultaneously. For example, a user can execute a program like MS Word to prepare a document and run another program (like Windows Media Player) to play songs. Both of these tasks can be performed simultaneously.

Today most of the modern Operating Systems running on personal computers belong to this category.

➢ **Objective :**

- The primary objective of the Time Sharing Operating System is to provide better response time to users.

➢ **Advantages :**

- Like Multiprogramming OS, Time Sharing OS also significantly improves **system throughput** and **resource utilization**.
- As response time is low, it also allows sharing of computers among multiple users simultaneously. It gives an illusion/impression to each user as if the entire computer resources are available to him solely.

➢ **Required Features :**

Here, as multiple programs reside simultaneously in the main memory, Operating System requires the following features:

- To store several programs in memory simultaneously, better **memory management** & **protection** are required (Chapter 3).
- As memory is limited, concepts like **swapping** and **virtual memory** are used to accommodate more than one program in memory. (Don't worry. These topics are explained in chapter 3.)
- CPU **scheduling** (chapter 2) should be sophisticated to provide fairness to all programs.
- Disk management and **file systems** (chapter 4) are complex now.
- **Synchronization & Communication** among running programs and problems like **deadlock** should be overcome (chapter 2).

[NOTE :

- Various Operating Systems, like Multiprogramming, Time Sharing, Multitasking, Multiuser, and Multiprocessing, have no clear boundaries to separate them from each other.
- Different authors have different opinions regarding these terms. So, it is possible to have a slightly different description of these terms from other sources.]

1.3.4 Multithreading Operating System

Multithreading Operating System supports the concept of multiple threads within a single process environment.

Thread is a part of a process that can be executed independently from the other parts of a process. A thread is also considered a flow of control within a process. A process that contains multiple threads, i.e., multiple independent flows of control, is referred to as a multithreaded process.

Generally, an application is implemented as a separate *process* with several *threads* of control. Sometimes, a single application needs to perform more than one task simultaneously. For example, a word processor needs to accept input, provide spell-checking, and auto-saving. But how are all these possible at the same time ?

There are different threads in a single process that perform these different tasks simultaneously. For example, one thread may accept inputs, while the other may provide spell-checking, and the third one may do auto saving. And all of these execute simultaneously.

The operating system, which supports such functionality, is called a multithreading operating system.

Such systems provide advantages like Good response, Resource sharing, Better economy and Utilization of multiprocessor architectures.

1.3.5 Multi-user Operating System

Single-user Operating Systems allow a single user to access a computer system at a time. Multiple users cannot use a single computer at a time. For example, this situation can arise in a lab or at home. You want to use the computer, but someone else is currently using it. So, you must wait for that user to finish before using the computer system.

In contrast, Multi-user Operating Systems allow multiple users to access a computer system simultaneously.

Access to the computer system is usually provided via a network so that users access the computer remotely using a terminal or other computer.

A terminal contains only I/O devices, such as a keyboard and monitor. It is used to provide interaction between users and computer systems. A touch-screen help desk provided at a railway station or an ATM provided at a bank is an example of such a terminal.

Multiple users can access a single computer system through time-sharing and multiprogramming. Here, a CPU is time-sliced at a regular interval among various users. This gives an illusion/impression to each user as if the entire computer resources are available to him solely. (Remember, it's only an illusion...!!!)

UNIX, VMS and mainframe operating systems, such as *MVS*, are examples of multi-user operating systems.

This kind of Operating System is much more complex than Single-user Operating System. The operating system must make sure that –

- Each program executed by multiple users has a sufficient and separate resource.
- Resources, such as printers, should be shared in a fair and proper way.
- Requests from one user should not affect the operation of other users. For example, a request to print a document should not affect the printing operation of a different user.
- Only authenticated and valid users should use the data stored on computers.

Expensive hardware can be shared among several users using multi-user Operating System. This provides better utilization of resources.

One problem with a Multi-user system is that as more users access it, the performance becomes slower and slower. Also, another disadvantage is the cost of hardware and software, which tend to cost more than single-user operating systems.

1.3.6 Multiprocessing Operating System

A Multiprogramming Operating System allows more than one program to run concurrently on a single processor system. But, a single processor cannot execute more than one instruction at a time. So, the CPU is scheduled *rapidly* among several programs running concurrently. This provides a pseudo-parallelism and gives the illusion that all programs are running in a parallel manner.

True parallelism is only possible if multiple processors are used to execute programs. Examples of computer systems having multiple processors are –

- Dual-processor personal computers
- Powerful servers containing many processors, and
- Cluster of workstations

The Operating System that supports multiple processors is referred to as Multiprocessing Operating System.

So, a Multiprocessing Operating System allows more than one program to run concurrently on a multiprocessor system.

➢ **Advantages :**

- **Increased Throughput :**
  - Multiprocessing provides true parallelism. So, by increasing the number of processors, more and more tasks can be performed simultaneously. This results in increased system throughput.
- **Increased Reliability :**
  - If tasks are appropriately distributed, the failure of one processor does not result in a complete system failure. Instead, it only slows down the performance. This results in increased reliability.
- **Economy of Scale :**
  - Economically, multiprocessor systems are better than multiple single-processor systems. A multiprocessor system can share resources among multiple processors. So, if several programs operate on the same data set, it is better to store those data on a disk and allow multiple processors to share it; rather than using multiple single-processor systems.

➢ **Types :**

A Multiprocessing Operating System can be classified into two categories based on the architecture of the computer system, as explained below:

i. **Asymmetric Multiprocessing (AMP)**

- Here, as shown in Figure 1.5, one processor works as a master and controls the entire system. Operating System is executed on this processor. This processor identifies the user tasks to be performed and distributes them to other processors.
- Other processors work as slaves. They execute programs assigned by the master processor or have some predefined tasks.
- This is also referred to as the Master-Slave relationship.
- **Disadvantages :**
  - Failure of the master processor results in failure of the entire system.
  - The Master processor has to cater to all other slave processors. So, it is possible that the master processor may be overloaded while slave processors may remain idle. This results in a bottleneck for the performance.

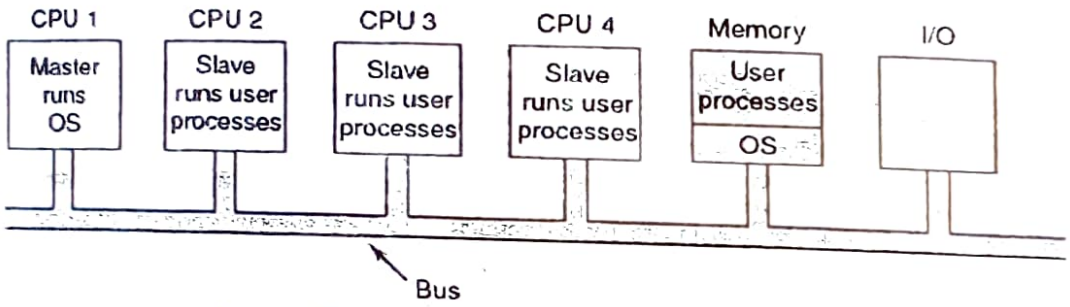

Figure 1.5 : Asymmetric Multiprocessing Architecture

ii. Symmetric Multiprocessing (SMP)

- As shown in Figure 1.6, all processors are considered peers (or identical); no master-slave relationship exists between processors.
- Each processor executes an identical copy of the Operating System. This Operating System controls the working of processors and provides functionalities required for multiprocessing, such as scheduling, load balancing, and memory management.
- This system overcomes the problems of asymmetric multiprocessing.

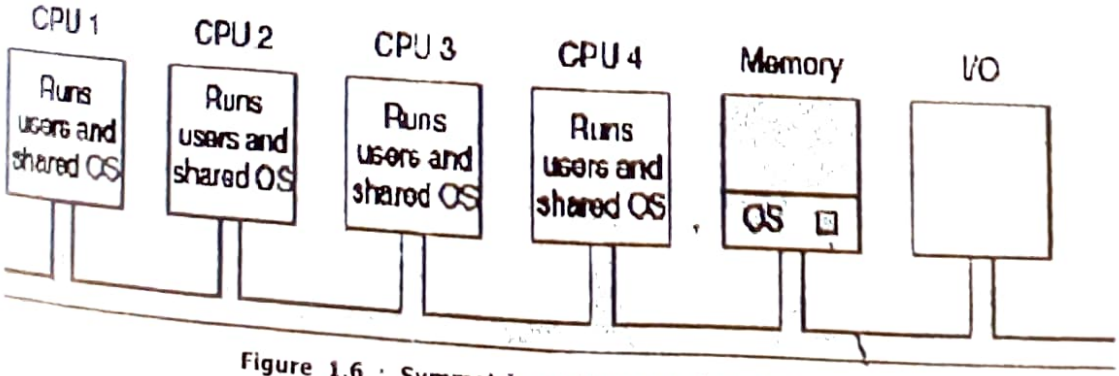

Figure 1.6 : Symmetric Multiprocessing Architecture

1.3.7 Real-Time Operating System

Real-Time Operating Systems strictly require processing input data in a pre-specified time duration. In such types of Operating Systems, time is the crucial parameter.

Here, input immediately affects the output. Time is very critical.

Such characteristics include systems to control nuclear power plants, oil refining, air traffic control systems, air defense systems, etc.

In all such systems, input coming from some sensors should be processed immediately. The results of such processing affect the output, which generally controls some devices. All these operations must occur within some time limits.

- Types
  - **Hard Real-Time System :**
    - All critical tasks *must* get completed strictly within the specified time limits. In other words, tasks are guaranteed to occur in time.
    - Example: system which controls the operation of an oil refinery.
  - **Soft Real-Time System :**
    - Less strict. Missing an occasional deadline is acceptable.
    - Example: Digital audio or multi-media system.
- **Required Features :**
  - As the primary objective is to provide a quick response to external events, **scheduling** should be priority based preemptive.
  - Most of the processes are memory residents for quick access. So, **memory management** is simpler.
  - I/O event management should be time critical. Like processes, databases and files are also memory residents for quick access. So, **file management** is also simpler.

1.3.8 Network Operating System (NOS)

A network operating system allows a set of autonomous computers, interconnected by a computer network, to be used together in a convenient and cost-effective manner.

In a network operating system, the users are aware of the existence of other machines. They can access remote resources by either logging into the appropriate remote machine or transferring data/files from the remote machine to their own machines.

Each computer has its own private operating system, and each user normally works on his/her own system. To access functionalities/resource of another system, a user requires to log in remotely. TELNET provides such type of facility to login into the other system.

Users are typically aware of their file locations and need to copy them before use through explicit file transfer commands. FTP (File Transfer Protocol) provides such type of facility.

These are tightly coupled systems. Processors share memory and clock.

Resources like printers and scanners can be shared using NOS.

NOS is not different from a single processor OS, but they require NIC (Network Interface Card) and drivers to provide networking functionalities. Also, NOS, such as Windows 2000 and Novel Netware, are not true Multi-user Operating Systems. Instead, they can be considered Single-user Operating System that supports networking.

1.3.9 Distributed Operating System (DOS)

Distributed Operating System distributes the computation (work of processor) among many processors.

The key objective of DOS is transparency. Here, users are not aware of the existence of other machines. Also, they do not need to be aware of the file locations to work with them.

When there is a higher load on one processor (computer), this OS automatically distributes the computation to other machines where the processor is idle or has a low load. Thus, load balancing/sharing is used here.

To the user, this system appears as a traditional uni-processor system. Users do not become aware of where their programs are running or where their files are located.

These are loosely coupled systems. Processors have their own memory and clock.

This system is more sophisticated and complex than NOS.

- **Advantages :**
  - **Resource sharing :**
    - Resources like processors are shared automatically here.
  - **Computation speed up :**
    - As the work of one processor is distributed among many processors, the complex task can be completed in less time.
  - **Reliability :**
    - If some part of the system fails, the entire system does not stop functioning.

1.4 **OPERATING SYSTEM SERVICES

Operating system services can be described from two different points of view:

- User point of view, where the primary goal is convenience, i.e., to make programming tasks easier; and,
- System point of view, where the primary goal is efficiency, i.e., to make efficient system operations.

These services are described below:

1.4.1 Services from the User Point of View

i. Program execution :

The primary purpose of OS is to provide an efficient and convenient environment for executing programs. So, an OS must provide various functions for loading a program into the main memory, executing it, and, after execution, terminating it.

ii. I/O operations :

A running program needs I/O operations to read input data and output the results. This I/O may be with a file or with a device. As users can't control I/O devices directly for security reasons, OS provides services for I/O operations.

iii. **File system manipulation :**

OS provides file system manipulation functionalities like creating, reading, writing, and deleting a file.

iv. **Communications :**

In a multitasking environment, more than one process is running simultaneously. Sometimes there is a need to exchange information among processes. Such processes may be on the same machine or different machines. OS provides mechanisms for such inter-process communications.

v. **Error detection :**

Errors may be in user programs or hardware like I/O devices. OS detects such errors and makes the user aware of them. It also provides some error recovery mechanisms.

1.4.2 Services from a System Point of View

i. **Resource allocation :**

When multiple users are sharing the same machine or when multiple jobs are running simultaneously, they need a fair allocation of resources among them. OS does this.

ii. **Accounting :**

Accounting is the process of keeping information about which user uses which resource and for what duration. Such information can be used to bill the users in a multi-user environment or to get usage statistics for future planning.

iii. **Protection :**

OS ensures that all access to system resources is controlled. Also, security from outsiders is essential.

1.5 LINUX OPERATING SYSTEM

Linux was first developed by Linus Torvalds, a student in Finland, in 1991. But now, Linux is not owned by anyone. No one company or individual "owns" Linux, like Windows is owned by a single company Microsoft.

Linux is the most famous *free and open source* Operating System.

Open source Operating Systems are available in source-code format rather than as compiled binary code. Anyone who receives it is free to make changes and redistribute it. You can change the source code of Linux as per your requirements. Not only this, but you can also distribute this modified OS. You need to be concerned that you have to provide your modified source code with this OS.

Many people have done such work, and as a result, many flavors of Linux are available in the market. Some known ones are Red Hat Enterprise Linux and its derivatives, such as Fedora and CentOS, Debian and its derivatives, such as Ubuntu and Linux Mint, Linspire, and PCLinux.

Day by day, Linux is becoming famous. It uses most of the features of UNIX. It is a multi-user, multitasking Operating System that also supports a handy Graphical User Interface (GUI). It is robust and secure. You rarely hear about virus attacks on Linux systems, as with Windows.

Linux can be found on many devices – from personal computers, mobiles, tablets and embedded systems to mainframe computers and supercomputers.

This section introduces the Linux Operating System.

1.5.1 History of Linux

In 1970, Ken Thompson and Dennis Ritchie at AT&T Bell labs developed UNIX operating system. Earlier, this system was known as UNICS (UNIplexed Information and Computer Service), but later it became famous as a UNIX. In 1973, it was rewritten in the 'C' programming language (a higher-level language). This made UNIX the world's first portable operating system, capable of being easily ported (moved) to any hardware. UNIX became easy to understand and modifiable. This was a major advantage for UNIX. This step led to the wide acceptance of UNIX among various users at that time.

In 1987, Andrew S. Tanenbaum created MINIX operating system. MINIX was a Unix-like operating system based on a microkernel architecture. MINIX was a text-oriented operating system with a kernel of fewer than 6,000 lines of code

In 1991, Linus Torvalds created the Linux kernel. He was a student at the University of Helsinki, Finland. He wanted to use UNIX on his own PC. But free versions of UNIX were too old at that time. Also, other commercials were too costly for him. So he thought of developing his own code. He researched UNIX and MINIX. He wrote a program on MINIX using the GNU C compiler. Finally, Linux was released on September 17, 1991. Linus made it open source. (Gem...!!!)

Between 1991 and 1995, Richard Stallman started the "Free Software Movement" and created the GNU project (collection of free software). Linux itself is a kernel and not an operating system. The collaboration of Linux and GNU gave us the "Linux" or "GNU/Linux" operating system. Some companies and open source communities adopted GNU/Linux codebase, did some modifications, and created their own version or distributions. Examples are: RHEL (Red hat), Fedora, Debian, Ubuntu, CentOS, Kali Linux

Note that Linux is a kernel, not OS. Linux is not a UNIX clone. It was written from scratch. A Linux distribution is the Linux kernel and a collection of software that together creates an operating system. But from now onwards, as in general use, Linux will be considered an operating system.

1.5.2 Features of Linux

As an operating system, Linux contains all the features that any operating system should have. Also, Linux is a UNIX-like Operating System. So, it contains most of the features of UNIX too. In addition, Linux contains some unique features that make it so popular.

These features are as given below:

1. **Free and Open Source Software :**
   - Open source means Linux is available with its source code. And, free means users have the freedom to make changes in source code according to their requirements. These modified versions can also be redistributed.
   - Also, most Linux flavors are either totally free or cost very little compared to other Operating Systems. (For example, Windows OS...!!!)
2. **Flexibility in Usage :**
   - Linux can be used for high-performance server applications, desktop applications, and embedded systems.
   - Due to this flexibility, Linux can be found on various devices such as mobile phones, tablet computers, network routers, personal computers, video game consoles, and even supercomputers.
3. **A Multi-user System :**
   - Linux is a multi-user Operating System. This means it allows multiple users to work simultaneously on the same system.

- Users can log in from different machines into the same machine using programs like 'TELNET'.

4. **A Multitasking System :**
   - Linux is a multitasking Operating System too. It allows multiple programs to run simultaneously.
   - Among simultaneously running processes, one process is the foreground process. Users can interact with this process directly. At the same time, other processes are background processes. They execute in the background without requiring user interaction.
5. **High Performance and Reliability**
   - Linux provides high performance with minimum hardware requirements compared to other Operating Systems. No other Operating System is more stable and reliable than Linux. System crashes, hangs, and virus attacks are almost absent from the Linux world.
6. **The Building-block Approach :**
   - Linux uses the building-block approach to perform complex tasks.
   - It provides a few hundred commands, each of which can perform one simple job. Such simple commands can be combined using pipes and filters to perform complex tasks. Thus, the small-is-beautiful philosophy is implemented here. (Pipes and filters are described later in this chapter.)
7. **Flexible Interface :**
   - Linux supports both types of interfaces – GUI (Graphical User Interface) as well as CLI (Command Line Interface).
   - GUI makes the task of users easy and so makes the Operating System user-friendly. CLI provides more options and control to the user. For example, complex tasks can be performed by combing multiple commands or creating shell scripts.
   - Users can choose any interface according to their convenience and expertise.
8. **File System Support :**
   - Linux supports many file systems such as ext, ext2, ext3, ext4, XFS, JFS, etc.
   - Along with these file systems, it also supports file systems supported by other Operating Systems, such as NTFS, so that Linux users can also access files managed by those Operating Systems.
9. **Programming Facility :**
   - The Linux shell is also a programming language.
   - It supports all the programming features, such as variables, control structures, loops and so on.
   - These features can be used to develop shell programs called shell scripts. Such programs can control and automate many of the system's functions. (Shell scripts are described in detail in section 5.9).
10. **Online help :**
    - Linux provides an online help facility for all the commands. For this purpose, it provides a command named 'man'. By using this command, the user can have instant help on any command, system call, or file format.
    - Along with this, as Linux is a community-driven Operating System, many developers and distributors work on it, and there is vast support available on the internet.

These features make the Linux operating system popular among other operating systems.

1.5.3 The architecture of Linux / Components of Linux

The following Figure 1.7 depicts the Linux architecture. Linux architecture is also known as the layered structure of Linux. As Linux is a UNIX-like Operating System, its architecture resembles that of UNIX.

The various layers depicted in Linux architecture are as follows:

1. **Hardware :**
   - The bottom layer is the hardware.
   - It consists of various physical devices like CPU, memory, disks, monitors, and printers.
   - These devices provide various services. For example, printers are used for printout purposes.
2. **Linux Kernel :**
   - The next higher layer is the Linux Kernel. It represents the core of the Operating System. (A kernel is described next).
   - It manages all the underlying hardware.
   - It directly interacts with the hardware and provides user programs required services.
   - It hides the complex details of hardware also.
   - In short, it provides a simple interface between user programs and hardware.
   - The main services of an operating system include process management, memory management, file system management, I/O management, security and protection.

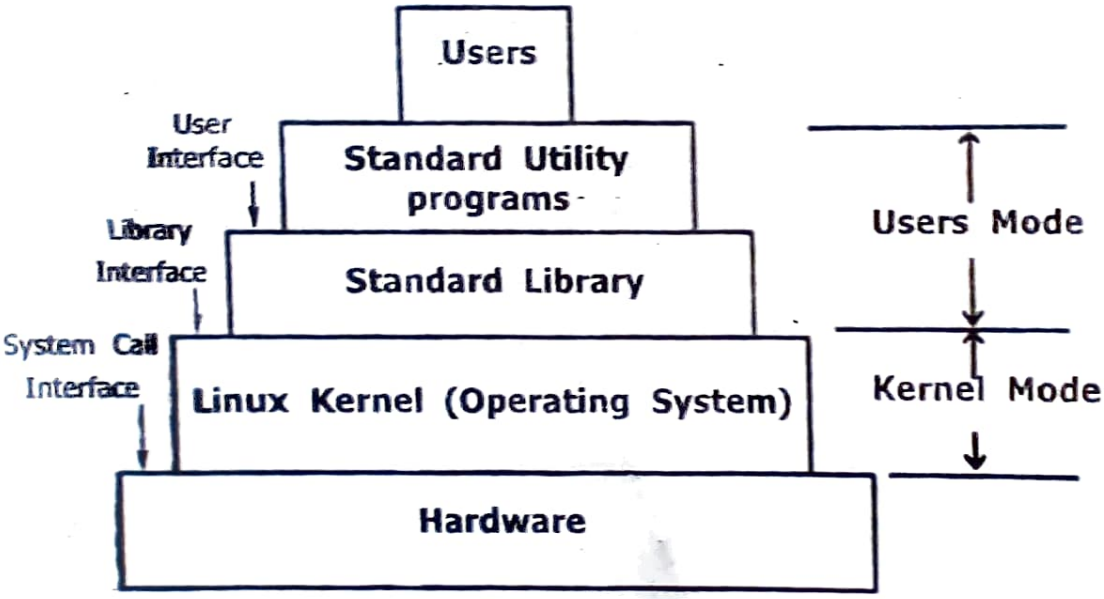

Figure 1.7 : Linux Architecture

3. Standard Library :
   - Above the operating system, the next layer is for the standard library.
   - It contains a set of procedures, one procedure per system call.
   - These procedures are written in assembly language and used to invoke various system calls from user programs.
4. Standard Utility Programs :
   - In addition to the operating system and system call library, all versions of Linux supply a large number of standard programs (application programs).

- Such programs include command processor (shell), compilers, editors, text processing programs, file manipulation utilities, various commands, graphical user interfaces and so on.
- Such programs make the user tasks simpler. Users interact with them, and they, in turn, interact with the operating system to get services from the operating system.

5. **Users :**

- The topmost layer is of users. User programs come in this layer. They interact with the system either by using library procedures to invoke system calls or by using utility programs such as a shell.

Here, some new terms came into the picture, such as **kernel**, **shell**, and **system call**. These terms are described below.

➢ **Kernel :**

- The kernel is the core of the Linux operating system. The kernel is a program that is loaded in memory when the system is turned on. It stays there and provides various services until the system is turned off.
- Linux uses a monolithic, modular kernel.
- Device drivers can be loaded and unloaded into the kernel in the form of kernel modules.
- Kernel interacts with the hardware directly. When a user program needs to use any hardware, it has to use services provided by the kernel. Special functions, called system calls, are used to request kernel. The kernel performs the job on behalf of the user process.
- In addition to providing services to user programs, the kernel also provides other services like process management, memory management, and file system management.
- In short, the kernel manages the entire computer system.

➢ Shell :

- The shell is an interface between the user program and the kernel.
- When the user logs in to the system, a process for the shell starts execution. It terminates when the user logs out of the system.
- Users can directly interact with the shell.
- It works as a command interpreter. It accepts commands from a user and translates them into a form that the kernel can understand easily.
- It is also a programming language. It provides various programming functionalities, such as looping and branching.
- Shell and its various concepts are described in detail in section 5.7.

➢ System Call :

- System calls are special functions.
- They are used to request the kernel to provide various services, such as reading from a file stored on a hard disk.
- They can be invoked via library procedures, via commands provided by a shell, or even directly from C programs in Linux.
- System calls are similar to user-defined functions. The difference is that they execute in kernel mode, having full access to all the hardware. In contrast, user-defined functions execute in user mode without direct access to the hardware.

1.5.4 Distributions of Linux

A Linux distribution is a collection of software applications built on top of the Linux kernel. It is also referred to as 'distro' for short. It represents the entire software package (Linux kernel + standard libraries + standard utility programs) from which Linux can be installed.

As Linux is a 'free to modify, use and distribute' Operating System, many companies, organizations and individuals have developed their own versions of the Linux Operating System. Some versions are more suitable to be used on servers, while some are more suitable to be used on personal computers. Some versions are designed to be used in embedded devices like mobile phones, while some are designed to be used on top of an existing operating system like Windows.

Different users may need to install different versions of Linux based on their requirements, so they need different Linux distributions. Some of the most popular Linux distributions are given below:

i) Linux distributions for desktop / personal computer:
   - Fedora
   - Ubuntu
   - Linux Mint

ii) Linux distributions for server:
   - Red Hat
   - Debian
   - OpenSUSE
   - Slackware
   - CentOS

iii) Linux distributions for virtual servers:
   - VM Ware
   - XenServer

A list of all kinds of Linux distributions can be found on www.distrowatch.com. Readers can decide which Linux to install after consulting that list.

1.6 **OPERATING SYSTEM STRUCTURES

We have seen that Operating System is one kind of software. And software is a collection of programs. And programs are a collection of instructions and data. All programs contain some kind of code (In the same way your C program contains the code). So ultimately, Operating Systems have some code as their internal structure.

This entire code can be managed in various ways. Structures of Operating Systems can be classified based on how this code is managed and how it provides functionalities. There are mainly four types of structures: Monolithic, Layered, Virtual Machines, and Microkernel/Client-Server based Systems.

These structures are described below:

1. **Monolithic Systems**

This is a very primitive form of OS. OS is written as a collection of procedures. Each procedure can call any of the other ones whenever it needs to.

Communication among procedures is provided by parameter passing and returning results, as used in function calling in C and other languages. Each procedure is visible to every other one. And, there is no any information hiding.

The basic structure is like this: there is the main program, which invokes service procedures. A set of service procedures carry out the system calls. And a set of utility procedures help the service procedures This is given in following Figure 1.8.

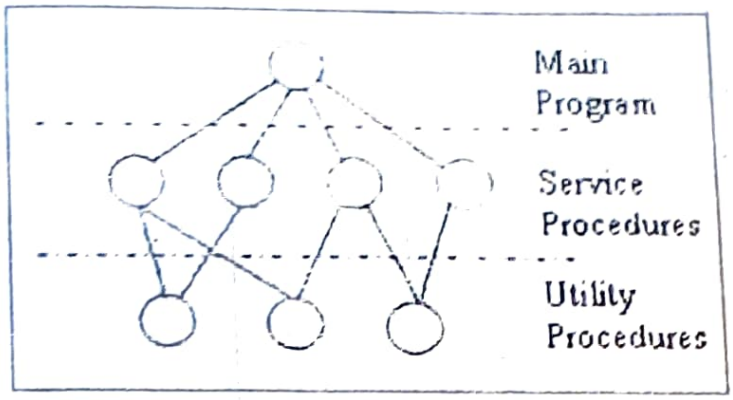

Figure 1.8 : Monolithic Structure

- **Example :**
  - DOS (Disk Operating System)
- **Advantages :**
  - No need for prior analysis; Efficient.
- **Disadvantages :**
  - Debugging is difficult; Also difficult to maintain and understand.

2. **Layered Systems**

OS is broken up into several layers (or levels). Each layer is built on lower layers. The bottom layer (layer 0) is the hardware layer. The top layer (layer N) is the user interface.

An Object-oriented approach is used here. Each layer uses functionalities only provided by the lower layers. Thus layer M can use the services of layer M-1. Likewise, layer M-1 can use services provided by layer M-2 and so on. Such a situation is given in Figure 1.9.

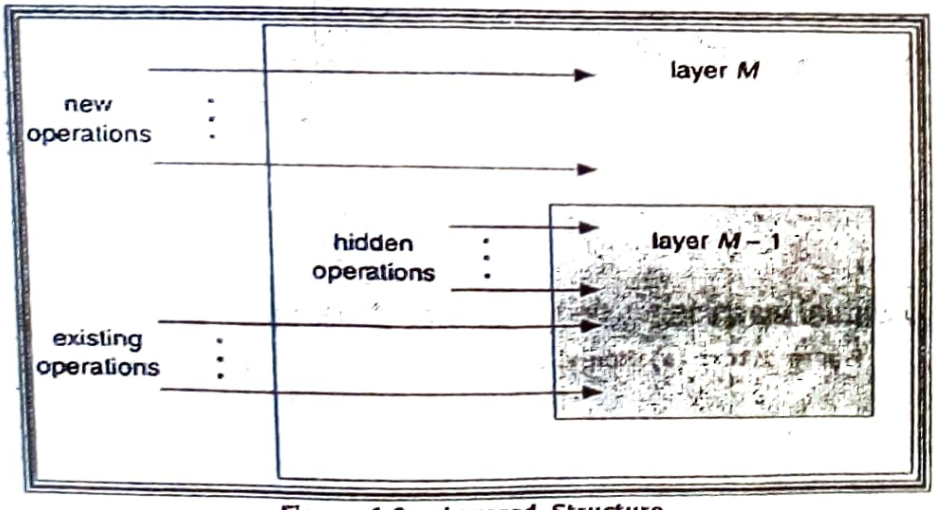

Figure 1.9 : Layered Structure

- **Example :**
  - THE operating system, OS/2 (descendant of MS-DOS)
- **Advantages :**
  - **Modularity** : System debugging and verification are simple.
  - **Information hiding** : Each layer hides its data and functions from the higher layer, thus, providing information hiding.
- **Disadvantages :**
  - **Prior analysis** of layers is required. One must determine to put which functionality in which layer.
  - **Less efficient**: each layer adds some overhead.

3. **Virtual Machines**

A **virtual machine** provides an interface identical to the underlying bare hardware.

Any multitasking OS provides – i) Multiprogramming and ii) an extended machine with a more convenient interface than actual hardware. A virtual machine completely separates these two functions.

Here, the 'virtual machine monitor' is the heart of the system. It runs on bare hardware, does multiprogramming, and provides several VM to the next upper layer.

Each process is provided with a virtual copy of the underlying hardware. Each process assumes that it has its own CPU, memory and other hardware. This is given in the following Figure 1.10 (b).

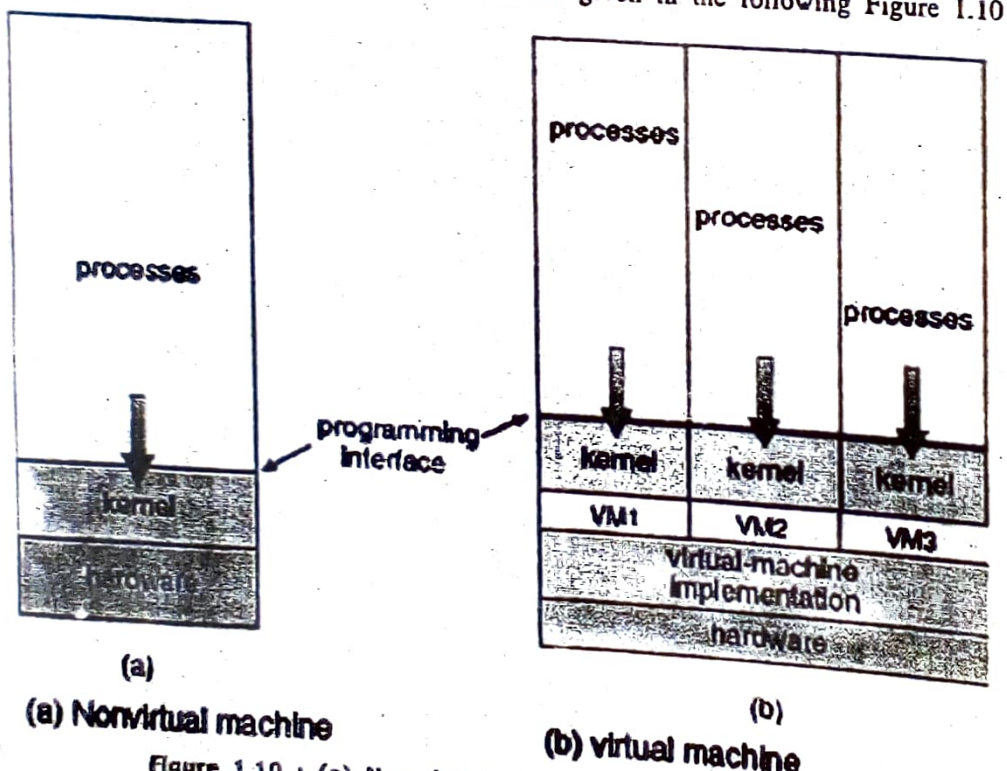

Figure 1.10 : (a) Non-virtual Machine (b) Virtual Machines

- **Examples :**
  - i) IBM's VM/370 ii) Java Virtual Machine

- **Advantages :**
  - **Security**: It provides complete isolation of system resources for protection.
  - **System development**: Any changes in a system can be done without disrupting normal system operations.
  - **Portability**: Provides portability across various platforms. Ex: Java virtual m/c.
- **Disadvantages :**
  - **Difficult to implement**: because it is difficult to provide exact duplication of the underlying hardware.

4. Microkernel / Client-Server based Systems

The operating system expands whenever new services/functionalities are added to OS. Consequently, the kernel (the core of OS) becomes large and difficult to manage.

We need to keep the kernel as small as possible to overcome this problem. So, code from the kernel is moved up to higher layers. All non-essential components are removed from the kernel and implemented as system and user-level programs.

As shown in Figure 1.11, when the user (client) process requires some service, such as reading a block of a file, it sends the request to a server process. The server process does the work and sends back the answer.

The primary function of the microkernel is to provide a communication facility between the client and server processes.

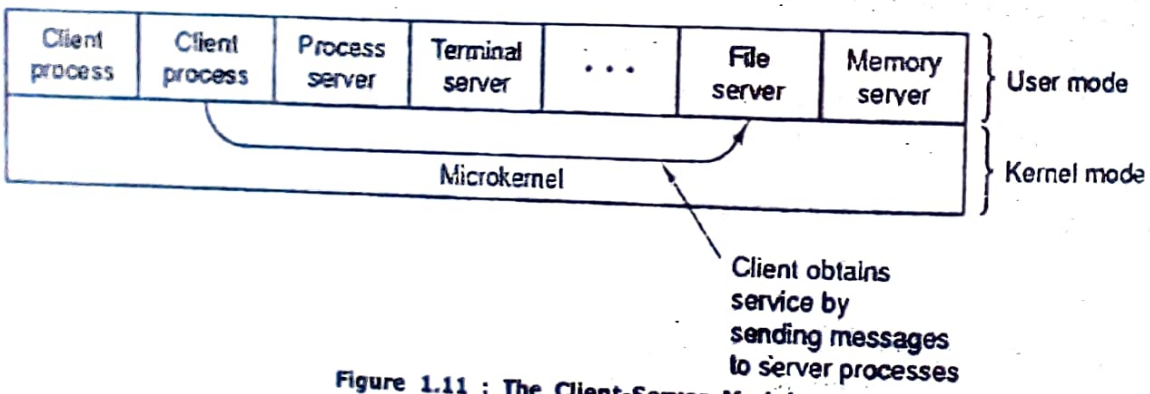

Figure 1.11 : The Client-Server Model

- Examples :
  - i) Digital UNIX, ii) QNX (real time OS), iii) Windows-NT.
- Advantages :
  - Ease of extending the OS: new services can be added to user space without changing the kernel
  - Security: two different modes - user mode and kernel mode provides security.
  - Reliability: failure of one service cannot crash the whole system.
  - Client-server model: can be adapted to use in Distributed OS.
- Disadvantages :
  - Overhead: communication overhead between user space and kernel space.

1.7 **PROCESSOR EXECUTION MODES: USER MODE V/S KERNEL MODE

Most recent CPUs operate in one of two ways: Kernel mode or user mode.

A bit in register PSW (Program Status Word) controls the mode of execution.

These modes determine the accessibility of the instruction sets and other hardware resources. All processors have their own set of instructions by executing which they get the required operations done. How many instructions can be executed by the CPU depends on the mode of the execution.

- **Kernel Mode :**
  - CPU can execute every instruction in its instruction set.
  - It can also access every feature of the hardware.
  - Operating Systems run in kernel mode, having access to the complete hardware.
- **User Mode :**
  - CPU can execute only a subset of its instruction set.
  - It can also access only a subset of the features of the hardware.
  - User programs run in user mode, having limited access. If they want some services from the hardware, they need to request the operating system. OS runs on behalf of the user programs and provides the required services.
- **TRAP :**
  - It is an instruction to change from user mode to kernel mode.
  - Occasions when this instruction is executed or when a mode is changed are:
    i) When system calls are executed,
    ii) When interrupts came, or
    iii) When exceptions are generated during program execution.

1.8 **SYSTEM CALLS

A system call is an interface between the operating system and the user programs.

All Operating Systems provide some set of system calls.

Let's take an example to understand the system calls and how they provide services. Consider a situation where an executing program requires some system service, such as reading data from a file. Suppose the file resides on the hard disk. As user programs run in user mode, they cannot directly access the hardware, here, the hard disk. So the program requests the operating system to provide service.

OS runs in kernel mode and can access all the features of the hardware. Here, OS runs on behalf of the user program and provides the necessary service, such as reading data from a file.

But how can a program request the operating system? System calls come into the picture here.

The actual mechanism of issuing a system call is highly machine-dependent. We may even require some assembly code to call them. So a **procedure library** is provided to make system calls from languages like C. This makes the programmer's task easier. (Have you ever used functions like `printf()`, `scanf()`, `clrscr()`, `getch()` provided by C language? These are the library functions. They make a call to system calls.)

➢ **Working of a System Call:**

System calls are performed in a series of steps. User programs generally call a library procedure (or library function). Library procedures execute TRAP instruction and change the execution mode from user mode to kernel mode. They also issue system calls. A system call is similar to some user-defined function, but the difference is that it executes in kernel mode, having full access to all the hardware.

The system call executes and performs the required operation, such as reading a block from the file. After its execution completes, the mode is changed again to user mode, and it returns to the library procedure. Now, the library procedure returns to the user program. And the user program proceeds with its execution.

1.9 SUMMARY

Operating System is an essential component of the computer system. Operating System works as a resource manager that manages all the resources and provides an extended machine for the user to work easily with the underlying complex hardware.

From the beginning, Operating Systems have evolved a lot. Batch operating systems don't provide user interaction and are useful in systems such as payroll. Multiprogramming OS tries to utilize the CPU optimally, while Time Sharing OS provides better response time along with utilizing the CPU optimally. Real-Time OS is used where time is the most critical factor in providing services.

Multiuser OS allows multiple users to access a computer system simultaneously, while multiprocessing OS supports multiple processors within a single computer system. NOS works with the networks, and DOS provides load balancing. Multithreading OS supports the concept of threads.

OS provides all the services to users to work with the computer system conveniently. Meanwhile, it also keeps watching the efficiency of the entire system.

Linux is the most popular free and open source Operating System. It is a multi-user, multitasking operating system right from its beginning. Linux has proved itself as one of the most powerful operating systems today. It can be found on devices ranging from mobile phones to supercomputers.

In Linux, work has been divided between kernel and shell. Shell interacts directly with the users, while kernel interacts directly with the hardware to access services from the hardware. Users interact with the kernel indirectly. For this purpose, they need either shell, C library procedure, or system calls. System calls are the special functions that run in kernel mode. The kernel runs in kernel mode having full access to all the hardware, while the shell runs in user mode having limited access.

Monolithic structures of OS are the old things...!!! Nowadays, most OS have layered structures. They are also extended to virtual machines and micro-kernels. Remember, OS is a complicated system, and structure *does* matter. Two execution modes protect system resources, and system calls are the way to receive services from the OS.

1.10 EXERCISES

1. Define Operating System. – OR – Define Operating System and give its goal.
2. What is an Operating System? Explain in brief.
3. Explain the need for Operating System. – OR – What is OS? What is the need for OS?
4. Explain the components of the computer system.
5. Differentiate: User-view of OS v/s System-view of OS.

6. Write a short note on: Operating System Services
7. List out services of an Operating System from a user point of view.
8. List out services of an Operating System from a system point of view.
9. List out types of the Operating System. – OR – Explain and list different types of Operating Systems.
10. Write a short note on: Batch Operating System. – OR – Explain Batch Operating System
11. Write a short note on: Multiprogramming Operating System. – OR – Explain Multiprogramming OS.
12. Write a short note on: Time Sharing Operating System. – OR – Explain Time Sharing OS.
13. Write a short note on: Multithreading Operating System.
14. Write a short note on: Multi-user Operating System.
15. Write a short note on: Multiprocessing Operating System.
16. Write a short note on: Real-Time Operating System. – OR – Explain Real-Time Operating System.
17. Differentiate: Multiprogramming OS v/s Time Sharing OS.
18. Write a short note on: Network Operating System.
19. Write a short note on: Distributed Operating System.
20. Write down the difference between Multiprogramming and Multithreading Operating Systems.
21. Differentiate: Time Sharing OS v/s Real-Time OS.
22. What is an Operating System? List types of Operating Systems & Explain Time Sharing Operating Systems.
23. What is an Operating System? List types of Operating System and explain any one of them in detail.
24. Explain Operating System Services.
25. Write a short note on: Linux Operating System
26. Provide an overview of the Linux Operating System.
27. Write a short note on: Features of Linux. – OR – Explain the characteristics of the Linux Operating System.
28. Explain: Linux architecture. Or, Explain: Linux layered structure.
29. Explain: Kernel, Shell and System calls.
30. Define: Kernel, Shell
31. List out various distributions of Linux.