

FUNDAMENTALS OF OPERATING SYSTEMS

A hand in a blue denim sleeve points upwards towards a large white circle. The background is dark blue with various technical graphics, including concentric circles, a radar-like pattern in the top right, and a vertical scale on the right side with numbers 0, 175, 270, 365, and 450. The title 'FUNDAMENTALS OF OPERATING SYSTEMS' is displayed in a white rounded rectangle at the top.

PROF SANJAY AGAL

Xoffencer

FUNDAMENTALS OF OPERATING SYSTEMS

Authors:

- Prof. Sanjay Agal

Xoffencer

www.xoffencerpublication.in

Copyright © 2023 Xoffencer

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through Rights Link at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13: 978-81-19534-89-0 (paperback)

Publication Date: 18 September 2023

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

MRP: ₹450/-



Published by:
Xoffencer International Publication
Behind Shyam Vihar Vatika, Laxmi Colony
Dabra, Gwalior, M.P. – 475110

Cover Page Designed by:
Satyam soni

Contact us:
Email: mr.xoffencer@gmail.com
Visit us: www.xofferncerpublishing.in

Copyright © 2023 Xoffencer

Author Details



Prof. Sanjay Agal

Prof. Sanjay Agal is working as Principal at Dr V R Godhania College of Engineering And Technology, Porbandar, Gujarat. He has received his PhD degree in Computer Engineering from Pacific University, Udaipur in 2016. He has published 16 international research paper and have two patents in artificial intelligence. He is a keen observer, planner & implementer with track record of developing operational policies/norms, systems & controls, motivational schemes & education standards for professionals during the career span.

Prof Sanjay Agal is a renowned computer engineer and an esteemed author in the field of computer engineering and technology. With a lifelong passion for computers and technology, Prof Agal has dedicated her career to advancing the field and imparting her vast knowledge to aspiring engineers.

Prof. Agal's journey in computer engineering began during her undergraduate studies, where she developed a deep fascination for the inner workings of computers and their transformative potential. Eager to explore the field further, she pursued a Master's degree in Computer Science, delving into topics such as programming languages, algorithms, and system design.

Driven by her thirst for knowledge and a desire to make meaningful contributions, Dr Agal went on to earn her Ph.D. in Computer Engineering. His doctoral research focused on developing innovative solutions for improving the performance and efficiency of

computer systems. His groundbreaking work in the field earned her recognition and accolades from leading academic institutions and industry experts.

Throughout her career, Dr. Agal has held prominent positions in academia. As a professor of computer engineering, he has mentored and inspired countless students, sharing his expertise, and guiding them in their pursuit of knowledge. His teaching style is known for its clarity, precision, and ability to distil complex concepts into easily understandable principles.

In addition to his academic endeavours, Dr. Agal has collaborated with industry leaders to implement cutting-edge technologies and address real-world challenges. His hands-on experience in the field has provided her with valuable insights into the practical applications of computer engineering principles.

Preface

The text has been written in simple language and style in well organized and systematic way and utmost care has been taken to cover the entire prescribed procedures for Science Students.

We express our sincere gratitude to the authors not only for their effort in preparing the procedures for the present volume, but also their patience in waiting to see their work in print. Finally, we are also thankful to our publishers **Xoffencer Publishers, Gwalior, Madhya Pradesh** for taking all the efforts in bringing out this volume in short span time.

Contents

Chapter No.	Chapter Names	Page No.
Chapter 1	Introduction	1-42
Chapter 2	Process And Threads Management	43-62
Chapter 3	Concurrency	63-71
Chapter 4	Process Communication	72-84
Chapter 5	Deadlock	85-94
Chapter 6	Memory Management	95-122
Chapter 7	I/O Management & Disk Scheduling	123-137
Chapter 8	Security & Protection	138-156
Chapter 9	Unix/Linux Operating System	157-168
Chapter 10	Virtualization Concepts	169-183

CHAPTER 1

INTRODUCTION

1.1 COMPUTER SYSTEM OVERVIEW

The emergence of the operating system as a software entity responsible for the management of hardware resources took place throughout the 1960s. Presently, the operating system is commonly regarded as a compilation of software programs that enable the operation and coordination of hardware components. An operating system may be defined as a comprehensive assemblage of software programs that are specifically developed to facilitate the efficient administration and synchronization of computer resources. There are several variants of operating systems, including UNIX, MS-DOS, MSWindows, Windows/NT, and VM. The comprehensive safeguarding of computer systems entails the implementation of software safeguards across several tiers.

Within the realm of an operating system, it is important to establish a clear distinction between kernel services, library services, and application-level services. These three categories delineate discrete partitions inside the operating system. Applications are performed by processes, which are interconnected via libraries that offer shared functionality. The kernel plays a crucial role in enabling development by creating a communication interface with peripheral components. The kernel is responsible for handling service requests that are initiated by processes, as well as managing interrupts that are created by devices.

The kernel, located at the nucleus of the operating system, is a meticulously crafted software intended to function inside a constrained state. The main responsibility of the system is to handle interruptions that arise from external devices, in addition to servicing requests and traps that are started by processes. In order to optimize the functionality of computer hardware, it is imperative to employ an Operating System that contains the capacity to recognize and establish connections with all hardware components, hence enabling users to effectively participate in productive endeavors. This part will mostly concentrate on the examination of the operating system, encompassing its progression and fundamental objective.

1.1.1 The Meaning And Purpose Of Operating Systems

An operating system (often shortened as OS) is a software program that is installed on a computer and serves the purpose of allowing the computer's hardware to connect and work in conjunction with the software that is installed on the computer. The following is a list of the key tasks that operating systems are responsible for:

- Reading data that is input via the keyboard
- Sending data to the display screen
- Managing files and directories
- Taking command of peripheral equipment such as printers and disk drives

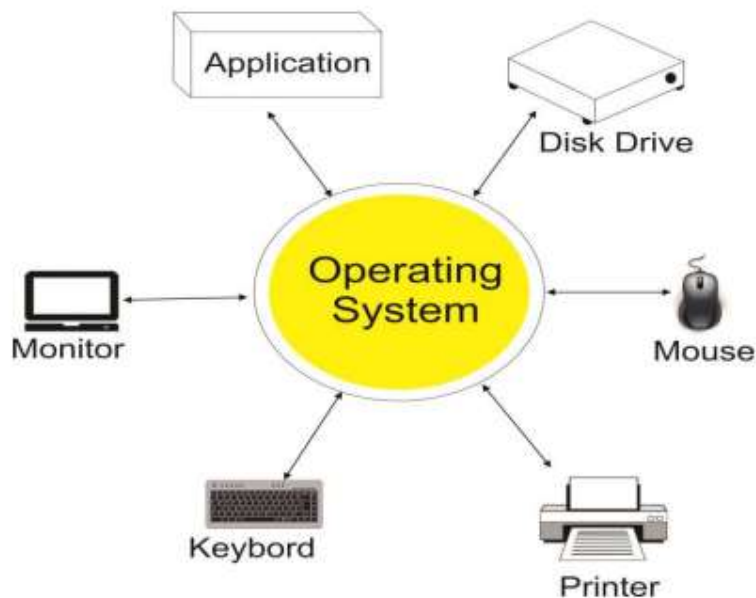


Fig 1.1 System software in conjunction with computer hardware

source: fundamentals of operating system, data collection and processing through by Er. Nishit Mathur 2015

The piece of software that is responsible for running the system is referred to as the operating system, and it is often stored on a medium of storage such as a hard drive,

CD-ROM, or floppy disk. When a computer is powered on, the operating system is transferred from the storage device into the main memory using the read-only memory (ROM) component of the machine. The storage device contains the operating system.

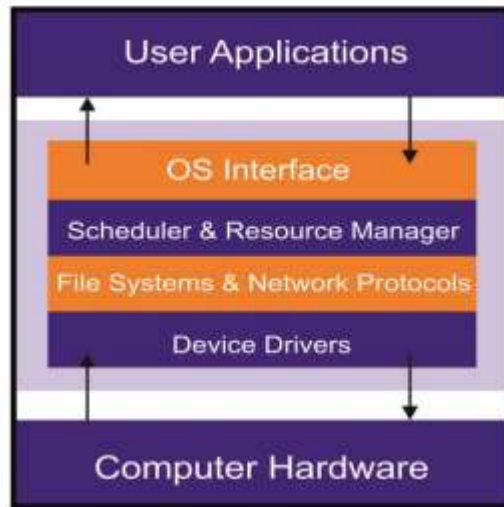


Fig 1.2 Operating System's Role in the Position

source: fundamentals of operating system, data collection and processing through by Er. Nishit Mathur 2015

An operating system is the component of a computer that is responsible for directing and managing the many tasks that are carried out by the system. It is in charge of managing the computer's hardware, directing the execution of application programs, and offering the users with a set of services to make use of. It is the user's interface with the computer system and serves as their point of contact. Users participate in indirect engagement with the operating system through the application programs, which serve as the channel for this interaction.

The following is a list of the responsibilities of various tasks that fall under the purview of the operating system: regulating the central processing unit directing operations in the random access memory

- Management of the data coming in and going out
- Management of the execution of the program

- Management of the files
 - Being in Charge of the Management of the Different Informational Parts That Comprise an Operating System
1. **The component that is local:** The term "kernel" refers to the section of the computer that houses the fundamental operating systems and programs. It is loaded into the main memory while the computer is in the process of starting up. It is in charge of a wide variety of responsibilities, all of which are kept in the main memory.
 2. **Portion of the whole that is comprised of non-residents:** When it is required to do so, this component of the operating system will be loaded into the main memory of the computer. It uses the Disk Operating System (DOS), which was originally developed by IBM and licensed to Microsoft.

Their work was focused on the creation of IBM's Operating System 2, also known as QS/2.

- The operating system known as XENIX or ZENIX, which was developed by Microsoft.
- WTNOWS was created by Microsoft and is responsible for its development.
- WINDOWS—NORTHWEST DIRECTION

1.1.2 The evolution of a number of different computer operating systems

The computer first made use of batch processing methods, which involve carrying out batches of work in rapid succession without halting for any intervals in between. After that, these programs are punched onto cards, which are subsequently replicated onto tape at the point where the processing was finally finished. The instant the computer had done the first job, it would immediately begin working on the second job mentioned on the tape. The tape included a list of jobs that needed to be completed. When properly trained operators began working with computers, they found that users dropped such tasks and ultimately returned to hold the outcome shortly after a given work had been completed. This led to the discovery that users dropped such jobs and eventually returned to hold the outcome. Users had a hard time with it since expensive

machines were necessary to assist in the processing of work of this sort. Consequently, users had a tough time.

In the late 1960s, when timesharing operating systems were first introduced, the transition from batch systems to timesharing systems for running applications got under way. Users found out that the Western Electric Teletype that was shown on the terminal functioned correctly while they were actively taking part in the process. As a result of the time sharing operating system, several users were allowed to utilize the computer at the same time, and each user finished each job in a fraction of a second before going on to the next task. It has been shown that a fast computer may operate on several separate user jobs at the same time, giving the appearance that the users were devoting their whole attention while the jobs were being completed. This discovery was made possible by the fact that fast computers are becoming more affordable.

Printing terminals made the discovery that the programs were constituted of a sequence of characters or command line user interfaces (CLI). Within these programs, the user was necessary to write answers in order to enter commands, which finally resulted in the instructions being scrolled down on study . Personal computers such as the Altair 8800 and the Personal Digital Assistant (PDA) were among the first computers to be used by individuals and businesses alike to perform monetary and organizational duties in the middle of the 1970s. In the year 1975, fans were given the opportunity to build their own Altairs by purchasing the computer as a kit. It did not come with an operating system due to the fact that the only controls it had are light-emitting diodes and toggle switches, which perform the input and output functions, respectively.

After some period of time had passed, users started attaching Altair's terminals and floppy disk drives to their computers. The year 1976 marked the debut of Digital Research's CP/M operating system for personal computers of this type. CLIs, which stand for command line interfaces, were a component of both CP/M and DOS in the future. These were quite similar to timesharing operating systems, in which only one user at a time had access to a single computer. The popularity of Apple's Macintosh computer, which was released in 1984, pushed the state of the art in hardware beyond its prior restrictions.

These limitations included a restricted display that was only black and white, as well as a restricted size. As time went on, multiple color Macs were being developed, and

Microsoft swiftly developed and released Windows as its graphical user interface (GUI) operating system. This coincided with the progression of hardware development. It was found out that the Macintosh operating system was built on decades of work on the subject of graphical user interfaces and applications for personal computers. Applications for modern computers need a single machine to accomplish a number of operations, and these applications may compete with one another for the resources that are available on the system. An very high degree of coordination is required for this, and it is possible that this can be controlled by a piece of software for controlling computer systems that is known as an operating system.

The kernel is the most fundamental part of an operating system, and its constituent parts include the File Manager, Device Drivers, Memory Manager, Scheduler, and Dispatcher. The kernel is also referred to as the "core" of the operating system.

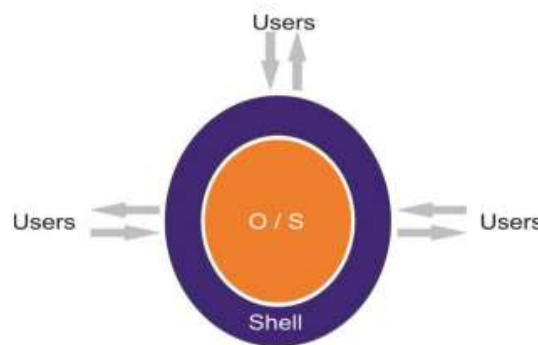


Fig 1.3 Interfaces of OS

source: fundamentals of operating system, data collection and processing through by Er. Nishit Mathur 2015

1.2 ARCHITECTURE,

1.2.1 An Explanation of Different Types of Operating Systems

The "Operating System" (OS) acts as a type of go-between for us and the many parts of the computer that are comprised of physical parts. It grants us access to a user interface, which makes it possible for us to run programs in a quick and efficient manner.

The architecture of an operating system is one of its most essential features since it pertains to the design and structure of all of the system's various components. Architecture is one of the most important characteristics of an operating system. These components comprise not only the central processing unit (CPU), but also memory management, file systems, device drivers, and the user interface. The architecture determines how these components interact with one another and how they provide services to applications and users. It also determines how they are provided by the architecture. The manner in which the components, on their own, perform services is also determined by the architecture.

Before we get too deeply into the Operating System Architecture, let's begin by acquainting ourselves with the fundamental vocabulary that is associated with operating systems.

1.2.2 The Centrum

The most essential component of an operating system is the kernel, often referred to as the core in some contexts. It is a very crucial part of the system that guarantees the dependability and security of the whole thing as a whole. The administration of processes and memory, as well as file systems and other functions, are all managed by the Kernel, which is responsible for a range of other tasks as well.

1.2.3 The encasing

Users are able to interact with an operating system by entering commands or running scripts through a user interface that is known as a shell. This user interface can either be a command line interface or a graphical user interface, depending on the needs of the application. This interaction may take place through the use of a command line or a graphical user interface, depending on your preference. It provides an interface that is more user-friendly and performs in a manner that is analogous to exposing the system services to the user.

1.2.4 The abbreviation CPU refers to the "central processing unit."

The central processing unit (CPU) of a computer is the primary component that is accountable for completing the bulk of the processing and computations. It is also known as the "brain" of the computer. The term "brain" is frequently used to refer to this component of the computer since it is the one that is in charge of carrying out the instructions that are included within the computer program.

1.2.5 Remembering things from the past

Memory is the word that is used to describe the components of a computer's hardware that are responsible for storing data, instructions, and information that the computer may access quickly. These components are known as RAM (random access memory) and DRAM (dynamic random access memory). There are several distinct varieties of memory that may be found in a computer, including the following:

- Random Access Memory, most commonly referred to as RAM: A computer's main memory is where it temporarily stores data and instructions and serves as the computer's primary memory. It is also referred to as RAM, which stands for random access memory.

- Read-only memory, commonly referred to as ROM, is a form of non-volatile memory that stores data and instructions that are essential for the computer to access when it is initially powered on. ROM is also known as read-only memory.
- Cache memory is a type of high-speed memory that is used to improve the overall performance of a computer by storing data and instructions that are often accessed and utilized. This type of memory is known as a "cache," and it is also abbreviated as "cache" for short.
- Virtual Memory is a way for managing memory that makes advantage of the capacity of the computer's hard drive to expand the amount of RAM, hence enhancing the computer's overall performance. This memory management technique is known as "virtual memory."

1.2.6 Exchange of information In the Interstices of the Processes

An other word for the interaction that takes place between processes is referred to as "interprocess communication." There may be several separate threads running concurrently within a process. Individual threads belonging to separate processes are able to communicate with one another within the kernel area. Ports are used to send messages to and receive them from the many threads running in the system. A variety of ports are present at the kernel level; the process port, the exceptional port, the bootstrap port, and the registered port are the most important of these ports. Each of these ports engages in some sort of conversation with the processes that are operating in user space.

1.2.7 Memory Management and Administration

Memory management is the process of allotting space in the main memory and managing the numerous operations that take place between the disk and the main memory. Memory management is sometimes referred to as memory administration. Nevertheless, there is also the development of virtual memory, which is something processes need to think about and account for.

When referring to the method of dividing and storing data, the phrase "virtual memory" is used to refer to the scenario in which a process demands more space than is currently available in the primary memory. After that, the various components of the operation are each saved in the main memory in turn until the CPU is ready to carry them out. This process continues until the CPU is ready to carry out the procedure.

1.2.8 There are Many Different Ways to Organize the System Architecture of Operating Systems

Operating systems are the basic component that support all modern computer devices, from mobile phones to servers. This includes any device that uses a computer. Nevertheless, not all operating systems are created equal, and the design of the operating system can have a significant effect on the performance, flexibility, and security of the system in a variety of significant ways.

In the following paragraphs, we shall study the many architectures that may be utilized for computer operating systems. In this section, we will discuss the fundamental ideas behind the creation of each architecture, as well as its advantages and disadvantages. Additionally, we will provide some examples of operating systems that make use of each technique.

1.2.9 Architecture Composed Of Only One Individual Block

The kernel and the device drivers are the key components of the monolithic operating system architecture's single module, which houses all of the system's activities and is the basis for the design of monolithic operating systems. Within the constraints of this architecture, the kernel region is where the operating system as a whole is responsible for carrying out its responsibilities. The kernel is in charge of the administration of all files, memory, devices, and processes, and it has direct control over these areas.

1.2.10 Benefits Associated with Employing a Monolithic Architecture

- **Swift:** Operating systems that are monolithic are characterized by their lightning-fast performance. As a consequence of this, they provide enhanced capabilities for the scheduling of activities, the management of memory, the management of files, and so on.
- **Every Component and the Kernel are Able to Communicate with One Another Directly:** Every single component and the kernel are able to communicate with one another in a way that is direct. In addition to that, it helps improve one's speed in a number of other ways.
- **Uncomplicated and Uncomplicated to Understand:** Due to the fact that all of its components are located in the same address space, its development is straightforward and uncomplicated, making it easy to understand and use.

It is more effective in terms of handling. Jobs of a smaller scale: Monolithic operating systems have superior performance when managing jobs of a lower scale. The monolithic operating system has the potential to lead to the production of errors and other faults inside the system. One of the drawbacks of monolithic design is that it has this limitation. This is because user programs and the kernel share the same address space, which is the reason for this behavior.

- **Difficult to Update:** A monolithic operating system makes it difficult to add or remove features from the operating system since all of the code for the operating system is concentrated in a single huge piece. This makes it difficult to update the operating system.
- **Not Portable:** The code that is produced in a monolithic operating system is difficult to carry with you or transfer to another system. This is due to the fact that the systems are incompatible with each other. This is owing to the fact that the entire code only works when it is moved as a single enormous chunk, and you will be need to relocate the entire thing in order for it to operate as intended.

1.2.11 The following are some examples of operating systems that are based on a monolith

The Microkernel Architecture is utilized by all three of these operating systems: Windows, Linux, and Mac OS.

The architecture of the microkernel ensures that the kernel contains just the most essential tasks, such as process control and memory management. In this configuration, the user space is in charge of managing not just the input and output but also the files themselves and their associated directories.

Advantages of a design based on a Microkernel • **Performance:** Due to the fact that a microkernel design is compact and self-contained, it has the ability to perform more effectively.

- **Protection:** Microkernels offer protection since they only incorporate those components into their make-up that are required for the system to continue operating properly. This eliminates the possibility of any faults occurring.
- **Scalable:** In contrast to a monolithic kernel, it possesses a plain and easy technique for growth.
- **Modularity:** Because Microkernels are modular, each of its modules may be altered, reloaded, or switched out without having any impact on the Kernel itself. This allows for more flexibility and customization of the operating system.

When compared with monolithic systems, distributed ones have a significantly lower number of incidents of system crashes. With the aid of the Microkernel interface, it may be possible to more simply create a system structure that is more modular.

Overhead in terms of performance: Overhead in terms of performance comes as a result of the necessity for inter-process communication between kernel modules. Using a microkernel design comes with a number of drawbacks, and this is one of them. When the drivers are implemented as procedures or processes, it is essential to either execute a context switch or make a call to a function. This is the case regardless of whether the drivers are implemented as procedures or processes.

Due to the restricted hardware support that it offers, the architecture of the microkernel may be more difficult to migrate to new hardware platforms. When compared to a monolithic system, the provision of services in a microkernel environment comes with a higher price tag. This is because of the increased complexity of the environment. The following is a list of operating systems that are some examples of those that are based on a microkernel:

The operating systems that are available are GNU Hurd, MINIX, QNX, and L4 microkernel.

Differences Between the Monolithic Kernel and Microkernel

Monolithic Kernel	Microkernel
In Monolithic kernel architecture, all system services, including device drivers, run in kernel space.	In Microkernel architecture, only essential services, such as scheduling and interprocess communication, run in kernel space. Device drivers run in user space.
Functions are tightly coupled and share a common memory space.	Functions are loosely coupled and communicate via message passing, which incurs some overhead.
System calls are fast, but errors in one component can crash the entire system.	System calls are slower due to message passing, but errors in one component do not necessarily crash the entire system.
Less secure because a bug or vulnerability in one component can affect the entire system.	More secure because if one component is compromised, the damage is contained to that component.
Examples include Linux, FreeBSD, and Windows.	Examples include QNX, L4, and MINIX.

1.3 GOALS & STRUCTURES OF O.S

During the process of putting an operating system into action, there are several different structures that might be used. The organization of the operating system is largely contingent on the method in which the many standard components of the operating system are associated with one another and incorporated into the kernel. This is because the kernel is the most fundamental part of the operating system.

An architecture that is known as an operating system provides the user application programs with the capability to interact with the machine's hardware in order to accomplish their tasks. Because of the intricacy of its architecture and the demand that

it be easy to use and customize, the operating system needs to be created with the utmost care in order to meet all of its requirements. The creation of the operating system through the use of supernatural methods is the primary strategy for achieving this objective. In order for these components to be regarded independent, each one of them needs to have its own unique set of inputs, outputs, and functions.

The following is a list of the operating system implementation structures that are discussed in this piece of research, together with explanations of how and why they function in the manner in which they do. In addition, the format of the operating system's framework is described in this section. Taking this into perspective, the components that make up the operating system are as follows:

1. Simple/Monolithic Structure
2. Micro-Kernel Structure
3. Hybrid-Kernel Structure
4. Exo-Kernel Structure
5. Layered Structure
6. Modular Structure
7. Virtual Machines

1.3.1 What is a System Structure for an Operating System?

We are searching for a framework that is uncomplicated and easy to understand because operating systems have such a sophisticated architectural design. Because of this, we will be able to make adjustments to an operating system so that it better meets our needs. Building an operating system by breaking it down into its component elements is a far more manageable task than building the entire system at once. A parallel may be drawn between this and the process by which we break down difficult problems into more manageable subproblems.

The operating system is considered to be one of the components of each and every component. One approach to think about the structure of an operating system is as a strategy that combines various components of the operating system into the kernel. This

is one way to think about the structure. As you'll see in the next section, putting up an operating system may be done in a number of different ways, and we'll go over them in more detail later.

1.3.2 Simple/Monolithic structure

These types of operating systems are simple, small, and limited; their architectures are not clearly defined. The distinctions between the various interfaces and levels of competency are not sufficiently made. The Microsoft Disk Operating System (MS-DOS) is one example of such an operating system. MS-DOS application programs may access the basic input/output functions of the operating system. With these operating systems, the entire computer will crash if one of the user apps stops working.

1.3.2.1 The advantages of a simple, one-dimensional structure

Such an operating system is easier for kernel developers to build, meaning that applications run better since there are fewer interfaces between the hardware and application software.

1.3.2.2 The disadvantages of a simple or monolithic construction

There are no clear boundaries between the modules, which adds to the structure's high degree of complexity, and data masking inside the operating system is not necessary.

1.3.3 The Micro-kernel's Structure

By removing any unnecessary kernel components and implementing those remaining components as distinct user apps and system programs, this structure builds the operating system. This causes the micro-kernel, a smaller kernel, to malfunction as a result. Since all new services may be added in user space rather than the kernel itself, the kernel does not need to be upgraded. This is among the benefits of this arrangement. Because of this, it is more secure and trustworthy because the operating system as a whole won't be impacted even if one of the services fails. As an example of this type of operating system, consider the Mac OS.

1.3.4 What are the advantages of using a microkernel structure?

Microkernels' small size makes it simpler to test them extensively, and this trait allows the operating system to be portable to other platforms. There are several disadvantages

to the microkernel architecture. If the level of communication between modules is raised, one may anticipate a decline in performance.

1.3.5 Framework Utilizing a Hybrid Kernel

Composed of a combination of the two, the hybrid kernel structure consists of both the micro- and monolithic-kernel structures. To put it briefly, it combines elements of the monolithic and micro-kernel techniques to provide a more sophisticated and advantageous technique. It preserves the flexibility and stability of micro-kernel structures while implementing the speed and architecture of monolithic systems.

1.3.6 The Advantages of a Hybrid Kernel Architecture

Because it combines the best features of both types of structure into one, it performs exceptionally well; it works with a wide range of computer hardware and software; it enhances the overall security and isolation of the system through the use of a micro-kernel implementation; and it increases overall system dependability by isolating critical functions into the micro-kernel for ease of debugging and maintenance.

1.3.7 Utilizing a Hybrid Kernel Structure Has Its Drawbacks

- It increases the overall complexity of the system by implementing both monolithic and micro structures, making the system more difficult to understand.
- Compared to a monolithic kernel, the micro-kernel's layer of communication with other components results in a reduction in performance and an increase in time complexity.

1.3.8 Configuration of the Exo-Kernel

An operating system called Exokernel was developed at MIT to provide application-level hardware resource management. The exokernel architecture separates the resource management function from the protection function to allow for application-specific customization. Exokernel sizes are often quite tiny due to the limited functionality they provide. Operating systems (OS) always impact the sort of apps that may be developed for them, as well as their functionality, performance, and reach, because they sit between software and hardware. The exokernel operating system takes

a direct approach to this problem, refuting the popular belief that an operating system should provide application developers with a set of abstractions upon which to construct their applications. The intention is to provide developers complete creative flexibility while imposing as few practical limitations as possible on their use of abstractions.

The following are some benefits of using Exo-kernel: it improves application performance; it separates management from security; it supports improved application control. Every user-space program is allowed to use its own custom memory management system.

- It is easier to test and create new operating systems.
- More efficient use of hardware resources is made possible by accurate resource allocation and revocation.

1.3.9 Among the disadvantages of the exo-kernel are:

- A decrease in consistency
- Exokernel interface architecture is a little bit intricate.

1.3.10 layered or tier-based arrangement

By decomposing an operating system into its constituent components, far more control over the system may be maintained. The operating system in this design is divided into several layers, sometimes referred to as levels. Layer N is the highest level, occupied by the user interface, while layer 0 is the lowest, occupied by the hardware. Each layer in the hierarchy makes use of the functionality of the levels underneath it thanks to the way these layers are organized. This simplifies the debugging process significantly since, once lower-level layers have been debugged, if an error occurs during the debugging process, it can only occur on that layer because the lower-level layers have already been debugged.

The main disadvantage of employing this structure is that data must be updated and passed on at every level. It results in an increase in the overhead that the system bears. Furthermore, since each layer may only use layers that are one step below it, the layers must be carefully arranged. This structure may be found in the UNIX operating system.

The several advantages of using a tiered structure:

- Operating system enhancements are easier to develop with layering since one layer's implementation may be easily changed without affecting the operation of the other levels.
- It is possible to complete system verification and debugging with comparatively minimal work.

1.3.11 Layered architectures are not without their drawbacks.

- When utilizing this structure as opposed to a simple structure, the application's performance is decreased. Because the upper levels only make use of the functionality of the lower layers, building the layers requires careful design.

1.3.12 Modular strategy or organization

It is frequently thought to be the best operating system approach. More specifically, a modular kernel has to be constructed. Only a limited number of necessary components make up the kernel; further services can be introduced at any moment during runtime or the boot process in the form of dynamically loadable modules. Since every kernel has specified and protected interfaces, it is comparable to a layered structure; yet, because any module can call any other module, it offers greater flexibility than a layered structure. This is because every kernel has protected and specified interfaces. The Solaris operating system, for example, is organized as shown in the image.

1.3.13 The term "VMs," or "virtual machines,"

The ability of a virtual machine to abstract our personal computer's hardware, including the CPU, RAM, disc drives, and Network Interface Card (NIC), into several independent execution contexts gives us the impression that every execution environment is a separate computer. This is completed in accordance with our specifications. A digital container would be one instance of this.

An operating system allows us to run several processes at once and make it appear as though each one is using a separate processor and memory by employing strategies like CPU scheduling and virtual memory. To do this, it is made to appear as though every task is utilizing a separate CPU.

The main issue with virtual machine technology is rooted in the utilization of disc storage technologies. Imagine a situation where the physical computer can only accommodate three disc drives, but it still has to be able to run seven virtual computers. It should be evident that attaching a disc drive to each virtual computer that is formed is not practicable since the software that creates virtual machines needs a significant amount of disc storage space in order to offer virtual memory and spooling. The answer is to provide CDs in digital format as an alternative.

1.4 BASIC FUNCTIONS

The most crucial task an operating system has to do is the allocation of resources and services. This role involves the allocation of memories, hardware, and central processing units, among other things.

- **Details:** Many practical applications, including a scheduler, input/output (I/O) programs, memory management, traffic controller, and file system, are included with the operating system. Together, these applications help manage the resources of the system.
- **Guaranteed security:** The operating system encrypts user data and uses a password to prevent unauthorized access to apps and user data, but we also need to install anti-malware software to protect the system from external threats.
- **Command and oversight of the system's functioning:** To help enhance the computer's performance, the operating system closely monitors the system settings. To gain a thorough picture of the system, it also records the duration of time between service requests and the computer's answer. Through the provision of crucial information needed for problem diagnostics, this can help improve performance.
- **Accounting for Employment:** Operating systems constantly keep track of how much time and resources are used by various users and tasks. One may use this data to assess how well a group of users or a single user is using the resources.
- **Tools for error-finding aid:** Through the constant system monitoring offered by operating systems, we are able to spot errors and keep computers from going down. coordination between many software programs and their individual users
Operating systems help the numerous users of computer systems by organizing and

assigning different software components, such as interpreters, compilers, assemblers, and others.

- **Memory Handling:** The operating system is in charge of the primary memory, sometimes referred to as main memory. A large number of words or bytes make up primary memory. Each word or byte has an address and may be accessed in turn. Because it is a highly rapid storage, the device's CPU, or central processing unit, may access it right away. An application cannot be executed until it has been loaded into the main memory. The operating system is in charge of the following tasks in order to manage memory efficiently:
 - It keeps track of the primary memory;
 - It assigns memory addresses and keeps track of RAM addresses that haven't been used by the application.

The operating system is in charge of deciding how long a process must run in multitasking mode and what order processes are allowed to access memory. When a process requests memory, memory is allocated to it; once the process is finished and the request is met, memory is deallocated.

- **Controlling the Processor:** In a multiprogramming environment, the operating system is in charge of controlling the order in which processes are granted access to the processor and figuring out how long each process must run for in order to be active. Another term for this is process scheduling. The following tasks are carried out by the operating system in order to control the CPU:
 - Preserves a log of the processes' present status.
 - To keep an eye on the current situation, utilize the traffic controller program.
 - When the central processing unit (CPU) is not required, it releases the reservations it has made for it.

1.4.1 Control of Equipment

Device connection will be managed by an operating system through the use of the proper drivers.

The following tasks are performed by the operating system in order to manage devices.

- Constantly keeping an eye on every gadget connected to the network.
- The input/output controller is the software that the operating system designates to oversee all of the linked devices.
- It controls the device itself as well as which process is granted access to what device and for how long. After that, it allocates the devices in a way that maximizes efficiency and effectiveness, de-assigning them when it decides they are no longer needed.
- Handling of Documents A file system is frequently arranged into directories to make browsing and usage easier. There are more directories and file types in these folders.

The operating system is in charge of the following file management-related tasks:

By collective noun, the facilities that comprise the file system are referred to as "the file system." It retains a variety of information, including the location of data storage, user access methods, and status.

1.5 INTERACTION OF O.S. & HARDWARE ARCHITECTURE

The term "kernel" refers to the piece of software that is considered to be the most fundamental component of an operating system on a computer. The term "kernel" refers to the piece of software that, in most instances, is in command of everything else that is included within the system. It is a part of the code for the operating system that is resident in memory at all times and is responsible for facilitating interactions between various pieces of hardware and software. This part of the code is referred to as the kernel. A complete kernel is one that mediates conflicts that arise between processes over such resources, manages all of the hardware resources (such as I/O, memory, and cryptography), and optimizes the use of shared resources (such as consumption of CPU and cache, file systems, and network sockets). After the bootloader has been initialized, the kernel is often one of the first programs to load on the majority of different types of computer systems. It is responsible for handling the remainder of the startup procedure in addition to requests from software for memory, peripherals, and input/output (I/O) activities. It does this by translating these requests into data-processing instructions for the central processing unit.

It is common practice to load the code of the kernel that is mission-critical into a separate region of memory, which is then protected from access by application software and other components of the operating system that are regarded as having a lesser level of significance. This secure kernel section is where the kernel is responsible for carrying out its obligations, which include regulating operating processes, hardware devices like the hard drive, and reacting to interrupts. These responsibilities include controlling hardware devices like the hard drive. On the other hand, application programs, such as web browsers, word processors, or players for music or video files, use a separate sector of RAM referred to as user space. This space is dedicated to the application applications. This separation prevents user data and kernel data from interfering with each other and producing instability and slowness,[1] as well as preventing programs that aren't working correctly from influencing other applications or putting the entire operating system to a standstill.

In addition, this separation protects user data and kernel data from interacting with one another and creating instability and slowness. Even on computer systems in which the address area that is accessible by programs includes the operating system kernel, memory protection is still used. This is done to prevent untrusted programs from making modifications to the core operating system. At the interface of the kernel, you will discover a low-level abstraction layer. When a process wishes to request a service from the kernel, the process must first make the necessary system calls in order to do so. These calls are often done by using a wrapper function in between them.

There are a few different architecture principles that may be applied to the kernel. When the CPU is in supervisor mode, monolithic kernels carry out their whole execution contained within a single address space. The fundamental motivation behind this action is to achieve greater levels of performance. The majority of a microkernel's services are carried out in user space, albeit not all of them are; this is analogous to how user processes operate. This is done mostly with resilience and modularity in mind as the end goals in mind. The MINIX operating system is a well-known illustration of a well-known example of a microkernel architecture. Instead, the Linux kernel is a single piece of software that has the ability to add and remove loadable kernel modules while the operating system is running. Despite this, the Linux kernel itself is a modular piece of software.

The major purpose of this fundamental component of a computer system is to provide the environment in which software programs may be run. It is up to the kernel to decide

which of the many programs that are now executing should be allocated to whatever CPU or processors are currently active at any one time.

1.5.1 Random-access memory

The term "random-access memory" (RAM) refers to a form of memory that may be used to store not just data but also computer instructions.[a] In the vast majority of instances, a program will not be able to correctly operate unless both of these components are available in memory. It is fairly commonplace for many apps to seek access to memory at the same time, which frequently causes the computer to request more memory than it has available. This can be frustrating for the user. It is up to the kernel to determine which parts of memory each process is permitted to use and what actions should be taken in the event that there is insufficient memory available to meet the requirements of all of the processes currently running.

1.5.2 A variety of input and output devices

The word "I/O devices" can refer to a number of different kinds of computer peripherals, such as keyboards, mice, disk drives, printers, display devices, USB devices, and network adapters, amongst others. Because the kernel often provides straightforward pathways for app access to various devices, the applications themselves do not need to be aware of the particulars of how these devices are really implemented. The kernel is responsible for abstracting these different methods.

1.5.3 management of the available resources

The process of managing resources begins with defining the execution domain, which is also referred to as the address space, as well as the protection mechanism that is used to arbitrate access to the resources that are included inside a domain. Both of these phases are necessary components of the overall process. These services are provided by kernels in addition to synchronization and inter-process communication (IPC) techniques. These implementations may be located within the kernel itself, or the kernel could rely on other processes that it is presently performing. Either way, the implementations could be considered part of the kernel.

Both of these outcomes are quite conceivable. Even if the kernel is necessary to provide IPC in order to allow running programs access to the facilities offered by each other, kernels are also required to provide running programs with a mechanism to make

requests in order to obtain access to the facilities. This is because IPC is required in order to provide running programs access to the facilities supplied by each other. In addition to this, it is the responsibility of the kernel to move the context amongst the many processes and threads that are running.

1.5.4 Memory management and administration

It is the responsibility of the kernel, which has unfettered access to the memory of the system, to ensure that other programs may access it in a secure way if this becomes required. The first step in the process of achieving this objective is typically the implementation of virtual addressing. Virtual addressing may frequently be achieved by paging and/or segmentation. The kernel is able to create the appearance, with the assistance of virtual addressing, that a certain physical address corresponds to another address, which is referred to as the virtual address. This is accomplished by simulating the mapping between the two addresses. There is a chance that every process will have its very own specialized virtual address space. This indicates that the memory that is accessible by one process at a particular (virtual) address may be distinct from the memory that is accessed by another process at the same location.

This is because virtual addresses are not physically mapped to physical memory locations. This prevents applications from crashing one another by allowing each program to operate as if it were the only one functioning (except from the kernel, of course). The kernel is the exception to this rule. On a wide variety of computer systems, the virtual address of a program may point to data that is not currently held in memory. This can happen in a number of different ways. Virtual addressing provides an extra layer of indirection, which enables the operating system to store information in locations other than the primary memory (RAM), such as the hard drive. This is made possible by the fact that it allows the OS to store information in areas other than the RAM. If it did not have this additional layer of indirection, the information would be required to stay in RAM at all times.

Because of this, operating systems are able to give permission for apps to utilize more memory than the system really has physically accessible to it. This is possible because of virtual memory. When a program calls for data that is not already present in RAM, the central processing unit (CPU) sends a signal to the kernel to let it know that this has happened. Following this, the kernel will respond by saving the contents of an idle memory block to disk (if this is necessary) and replacing them with the data that the

application requested. After then, the program may be resumed from the place where it was paused the first time, continuing from where it left off. The majority of individuals refer to this method by its more common term, which is demand paging.

In addition, virtual addressing makes it possible to establish virtual partitions of memory in two different areas of the storage space of the computer. One piece of the storage space is marked as "kernel space," while the other half is designated as "user space" for the applications that run on the operating system. Because the processor prevents applications from accessing kernel memory, it is impossible for an application to harm a kernel that is currently being processed by the computer. This fundamental segmentation of memory space has made a substantial contribution to the designs of true general-purpose kernels, and it is practically prevalent in systems that fall under this category. Nevertheless, particular research kernels, such as Singularity, make use of a variety of methodologies.

1.5.5 Administration of the machinery

In order for any essential actions to be carried out, processes need to have access to the associated peripherals of the computer. The kernel makes use of device drivers in order to exercise control over the many devices that are attached to the computer. When a piece of computer software encapsulates, monitors, and controls a piece of hardware (via the device's Hardware/Software Interface, or HSI), it is said to be acting on behalf of the operating system. This type of software is known as a device driver. It gives the operating system with an application programming interface (API), as well as techniques, information, and data on how to function and interact with a specific piece of hardware. In other words, it acts as a bridge between the hardware and the operating system.

Every single software and operating system requires device drivers in order to function properly, and this dependency is not just essential but also absolutely necessary. The abstraction of OS-mandated abstract function calls (also known as programming calls) into device-specific calls is the major emphasis throughout the design process of a driver. The primary objective of abstraction is to change OS-mandated abstract function calls. When combined with the proper driver, a piece of hardware should, in theory, perform as expected. Device drivers are necessary for the operation of a wide variety of hardware components, including video cards, sound cards, printers, scanners, modems, and network interface cards, among others.

1.6 SYSTEM CALLS

In the realm of computing, a system call is the mechanism by which a process requests the kernel of an operating system to provide it with a service that the process does not typically have permission to use. Generally speaking, the kernel of the operating system does not have authorization to offer the function that the process is requesting. System calls are the means through which a process can communicate with the operating system that it is running on top of. The vast majority of activities that interact with the system call for permissions that a user-level process cannot get to. For instance, the use of system calls is necessary for any kind of communication with other processes or I/O operations performed with a device that is already part of the system.

Using a method that is known as a system call, application software is able to communicate with the operating system in order to request a service. The application software is able to accomplish this thanks to this strategy. They make use of an instruction in the machine code, which causes a mode change in the central processing unit (CPU). Take, for instance, the change from supervisor mode to protected mode, which serves as an illustration. This is the section of the computer where the operating system is responsible for carrying out tasks such as gaining access to the memory management unit or hardware components. The majority of the time, an operating system will contain a library that can act as a go-between for the operating system and typical user programs. This kind of library is called a "intermediary."

It will almost always be a C library, like Glibc or the Windows Application Programming Interface (API). The management of low-level details, such as switching to supervisor mode and passing information on to the kernel, is the responsibility of the library. Some instances of system calls are closing and opening files, reading, waiting, and writing. In order for a process to be able to carry out any kind of useful activity, it has to be able to get access to the services that are provided by the kernel. Although every kernel has its own one-of-a-kind technique for achieving this, the vast majority of them provide either a C library or an API that can be accessed in order to call the relevant kernel functions.

The method of calling on the kernel function varies greatly from one kernel to the next, and there is no standard procedure. When memory isolation is being used, it is not possible for a user process to make a direct call to the kernel. This is due to the fact that doing so would be regarded a breach of the access control constraints that have been imposed by the processor. Some of the options include the following:

- By making use of a simulated interrupt within the program itself.
- Utilizing a system known as a call gate This strategy is supported by a very large percentage of hardware, which contributes to its widespread adoption. A call gate is an exclusive address that is stored in the kernel's memory at a location that is well-known to the processor. This location is called the call gate's starting point. This address is kept in a list that is updated regularly. Whenever the processor notices that a call is being placed to that address, it will instead place the call to the location that is intended to receive it. A violation of access will not occur as a result of this action. This cannot be accomplished without the aid of the hardware; nevertheless, the required hardware is not very difficult to obtain.
- By making use of a particular command in order to make a system call. This approach calls for the utilization of specific hardware support, which is something that most common architectural designs, most notably x86, can be lacking. However, in more modern varieties of the x86 CPU, system call instructions have been introduced. When these instructions are available, particular operating systems for personal computers will make use of them.
- Utilizing a queue that is kept in memory for processing. An application that makes a large number of requests but does not need to wait for the results of each request may add information about the requests they have made to an area of memory that the kernel scans on a regular basis in order to find requests. Because of this, the program is able to proceed without waiting for the outcomes of the requests.

1.7 BATCH

The work involved in doing jobs that are virtually identical to one another in a continuous loop again and again, but with the exception that in this case the input data is supplied for every iteration of the job, along with maybe the output file. Batch processing is the term that's used to describe this kind of labor. A batch operating system is a specific kind of operating system that, most of the time, calls for the use of a mainframe computer. The use of a computer of this kind began with the expectation that it would be able to handle considerable quantities of work that was recurrent in character. Through study, it has been discovered that a set of mainframe computers are

responsible for processing the pension statements of thirty million individual customers. After the initial commands were implemented, it was found that a batch task did not require any kind of contact from a human being. This was a significant finding. The process of setting a batch job is comparable to filling out a form in the sense that specific details need to be displayed in order to complete the activity.

1.7.1 The most fundamental database management systems

Batch processing is a technique that is employed not only for the purpose of automated transaction processing but also for the purpose of capable bigness database updates. This is in comparison to the methods of collaborative online transaction processing (OLTP), which are used for processing transactions online. Because it is primarily a batch approach, the extract, transform, and load (ETL) step of filling data warehouses is often carried out in batches in the majority of computerized systems.

1.7.2 Images

When working with digital images, it is usual practice to use batch processing to carry out a number of operations, such as resizing, converting, watermarking, or modifying image files in any other way. This practice is prevalent since batch processing is efficient.

1.7.3 Alterations of opinion

Altering computer files so that they are saved in a format different than their original one is a further use for the batch processing method known as "batching." One illustration of this would be a batch process that, for the sake of querying and displaying by end users, transforms legacy and proprietary file types to conventional and standard formats.

1.7.4 A time frame for batches

When the computer system is ready to begin batch jobs without interruption from online systems, there is said to be "a duration of less-intensive online assignment" continuing in what is known as a batch window. This is because the batch jobs require less processing power than the online systems. Jobs might be abandoned at any moment over the course of a 24-hour day due to the presence of a number of antiquated computer systems that were incapable of real-time processing and could only do batch

processing. As a consequence of the completion of the assignment processing, the online procedures may only be required between the hours of 9:00 a.m. and 5:00 p.m., with dual shifts being available for batch work; in this particular instance, the batch window would be sixteen hours.

The issue is often not that the computer system is unable to handle both online and batch processing at the same time. Rather, the issue is typically that the batch systems limit access to data in an integrated state, which is segregated from online changes until the batch processing is complete. For example, the computation of interest, the transmission of reports and data sets to other systems, the printing of statements, and the processing of payments are all examples of tasks that are considered to be "end-of-day" duties (EOD jobs) in a financial institution such as a bank.

1.7.5 The approach of processing that uses batches

A approach that is sometimes referred to as batch processing is a procedure in which a collection of information is organized in a batch before the processing actually gets underway. It is possible to draw parallels between the functioning of a batch, on the one hand, and an account of orders, on the other. It is feasible for the operating system of a computer to scramble the data by supporting a script or batch file, or it is conceivable for a system to utilize a macro or an inner scripting tool to scramble the data. Both of these methods are described in the following paragraphs. Utilizing punched cards as a means to enter information into early computers was the method of choice for doing this task. Batch processing is a sort of processing that took place in batches, which is where the term "batch processing" originates.

Each individual activity that has to be finished receives its own "job" in a processing system that operates in batches. The distribution of jobs is being done in order to speed up the process of achievement while also reducing the requirement for direct human engagement. Every entrance parameter is set up in advance with the use of scripts, command-line arguments, control files, or task control language, whichever is most appropriate. This is in contrast to "online" or "back-and-forth" apps, which ask the user for input that is analogous to what they have previously supplied. After receiving its input in the form of a number of fact files, a piece of software will proceed to process those files until finally generating its output in the form of a number of fact files. Because the input data are gathered into batches, also known as sets of records, and each batch is then handled as a single unit throughout the operation, this form of operation is referred to as "batch processing."

This is because each batch is processed as a single unit during the operation. There is now accessible an additional batch of the results, which may be used in the evaluation once more if necessary. Batch processing has been intimately connected with the usage of mainframe computers ever since the introduction of electronic computing in the 1950s. The use of early computers to do batch processing was necessary for a wide variety of reasons, all of which must be taken into consideration. According to one school of thought, the most urgent concerns that active firms were facing at the time in terms of both assessments of profitability and competitiveness were accounting challenges such as billing. This was the case in terms of both evaluations of profitability and competition. Billing may very effectively be considered as a batch-oriented business process, and all organizations are required to bill their customers in a dependable and timely manner. In addition, each and every computer resource had been expensive; hence, the sequential submission of batch jobs on punched cards was a solution that was compatible with the resource restrictions as well as the technical growth that was occurring during the time period in question.

In the years that followed, interactive sessions using linked text-based computer terminal interfaces or graphical user interfaces became increasingly prevalent. These interfaces may also be called graphical user interfaces. In addition to this, early computers' primary memories were not designed to hold several copies of the same software at the same time, which rendered them incapable of performing their intended functions effectively if they attempted to do so. Batch processing is still commonly utilized in the mainframe computing sector; however, practically all types of computers are now capable of executing at least some batch processing; even if it is just for "housekeeping" chores, batch processing may be performed.

These include personal computers (PCs) that run UNIX, as well as those that run Microsoft Windows, Mac OS X, and even operating systems for smartphones. As computers become more commonplace in society as a whole, it is quite unlikely that batch processing will lose some of its significance. One of the fundamental reasons why batch processing is still commonly utilized in the majority of firms is because it can be modified to a broad variety of regular business operations. This is one of the reasons why this approach is still widely used. However, batch approaches are renowned for the number of mistakes they produce. Although online systems are still able to function even when manual facilitation is not expected, there is no assurance that they have been optimized to coordinate the completion of a number of actions in a short amount of time.

As a consequence of this, even brand new systems typically contain one or more batch processing methods. These methods are used for the purpose of updating information at the end of each work day, creating reports, printing studies, and performing a variety of other non-interactive tasks that must invariably be completed consistently within specified timeframes for the purpose of conducting business. The modern batch applications that are used nowadays make use of contemporary batch designs such as Jem the Bee or Spring Batch, which are both written in Java. This is done so that the applications can provide the fault tolerance and scalability that is necessary for high-volume processing. Additionally, analogous frameworks for a variety of different programming languages are utilized as well.

Batch applications are generally connected with grid computing systems so that a batch job may be dispersed across a large number of processors in order to ensure high-speed processing. However, this may result in major programming conflicts. Batch applications are typically coupled with grid computing systems. Processing a huge amount of records in batches places particularly rigorous demands on the architecture of the system as well as the application. Modern mainframe computers, in conjunction with designs that provide robust input/output throughput as well as vertical scalability, have a propensity to deliver superior batch performance compared to other available choices. The usage of batch processing is at its most efficient when it is used to tasks that need the processing of a considerable amount of data on a recurring basis.

1.8 MULTIPROGRAMMING

A multiprogramming operating system enables a higher number of software applications to be run on a computer with a single processor than a traditional operating system. When using an operating system that supports many applications, the other programs are ready to use the central processing unit (CPU) until it becomes necessary for one of the programs to wait for an input/output transfer. Because of this, it's possible for several distinct jobs to share the time that the CPU is being used. On the other hand, it is not necessary for them to carry out the duties connected with their employment at the same time period.

Depending on the context, the execution of a piece of software may be referred to as a "Task," "Process," or "Job." When compared to systems that process information serially or in batches, processing information concurrently results in a reduction in the amount of system resources that are used and an increase in the amount of work that is

completed by the system. The primary goal of multiprogramming is to facilitate the management of all of the resources that are made accessible by the system. A multiprogramming system is comprised of the main components of a file system, command processor, transitory area, and input/output control system. As a direct result of this, multiprogramming operating systems have been developed, each of which is equipped with the capacity to store many programs in accordance with the subsegmenting of various sections of the transitory area. The essential activities of the operating system are linked to the techniques for managing resources in some kind or another.

The vast majority of multiprogramming operating systems may be classified as belonging to one of these two groups. Each of these is explained in more detail below:

1. An operating system that allows for the running of many programs at the same time
 2. An Operating System Designed for a Group of Users
- **System Operation That Is Able to Perform Multiple Tasks:** An operating system that supports multitasking makes it possible to run many software programs at the same time. This is achieved by the operating system, which loads and unloads each program into and out of memory in a sequential fashion according to the requirements of the task at hand. When a program is deleted from memory, a duplicate of it is copied to disk in order to ensure that it may still be accessed in the event that it will be required at a later time.
 - **System for the Management of Activities Carried Out by Multiple Users:** Several users are able to share the processing time of a single powerful central computer by using a multiuser operating system. This is accomplished by connecting to the computer through a number of different terminals. This is made possible by the capability of the operating system to quickly switch between terminals, with the end effect being that each terminal receives a certain amount of CPU time on the primary computer. It may appear as though every user has continuous access to the main computer due to the quick pace at which the operating system is updated from terminal to terminal. It is possible that the length of time it takes for the primary computer to reply will become more obvious when there are a significant number of individuals using a system like this.

The Operational Capabilities of the Operating System for a Number of Different Programming Languages. The multiprogramming system enables several users to work on their respective jobs at the same time, with the results being able to be saved in the primary memory of the system. When the central processing unit (CPU) is in idle mode and one program is doing I/O operations, it is feasible for the CPU to divide its processing time across a number of different applications.

When one application is awaiting an I/O transfer from the computer's hardware, there is always room on the CPU for another application to make advantage of it. As a consequence of this, a large number of apps are able to share processing time. Even if not all jobs are completed at the same time, the processor is likely working on a high number of jobs simultaneously. In addition, a piece of yet another process may be completed before going on to the subsequent step, and this pattern might continue indefinitely. As a direct result of this, the major goal of a multiprogramming system is to keep the level of activity of the CPU at a consistent level until there are some jobs available in the job pool. Because of this, a computer with a single processor is capable of executing a huge number of programs, which ensures that the central processing unit (CPU) is never left idle.

- **The following are a few examples of operating systems that support many programming languages:** There are many various kinds of multiprogramming operating systems, some examples of which include the capacity to download programs, the capability to transmit data, MS-Excel, Google Chrome, the Firefox browser, and a huge variety of other applications. Other examples include the Windows operating system, the UNIX operating system, and other kinds of microcomputers including XENIX, MP/M, and ESQview. Both of these systems are instances of operating systems.

The benefits and drawbacks of using a programming environment that allows for the creation of a number of different programs at the same time. The multiprogramming operating system has a lot of advantages, but it also has a few disadvantages to consider. The following is a list of some of the positive aspects, as well as some of the negative aspects:

- **Additional Advantages:** Users of the multiprogramming operating system get a variety of different benefits as a result of using this system. The following is a list of some of the benefits that can be gained by this:

1. As a consequence, there is a reduction in the total amount of response time.
 2. Having the ability to conduct many tasks simultaneously within the same program may be quite handy.
 3. It contributes to the maximization of the computer's overall job throughput, which is a very significant advantage.
 4. The multiprogramming system may support several users at the same time in a single instance.
 5. Work that is contracted for a shorter period of time may be finished in less time compared to those that need longer hours.
 6. It is possible that it will help to improve the turnaround time for projects that only require a short period of time.
 7. It never stops functioning and contributes to more efficient usage of the central processing unit.
 8. The existing resources are utilized in an effective and efficient manner.
- **Inconsistencies in the use of terminology:** Users need to be aware of the many limitations that come with using an operating system that supports several programming languages. The following is a list of some of the potential drawbacks to consider:
 1. It is exceptionally challenging to comprehend, and it is quite sophisticated.
 2. It is necessary to schedule the activities of the CPU.
 3. Memory management is a vital component of the operating system due to the fact that the main memory is utilized to hold a variety of different jobs.
 4. The more challenging task is the one that requires you to manage all of the processes and obligations at the same time.
 5. Job seekers who are interested in permanent work may have to wait in line for some time if there are a large number of vacant positions.

1.9 MULTITASKING

A term used in modern computer systems that refers to the process of carrying out several tasks all at once. It is a logical extension of a multiprogramming system that enables the execution of a number of programs concurrently and is used to facilitate the execution of many programs simultaneously. With the help of this addon, it is possible to run a number of different applications all at once. A characteristic of an operating system known as multitasking is the capacity to let a user carry out more than one computer task at the same time. many activities can be grouped together and referred to as a process. This allows many jobs to utilize the same resources for processing, such as a central processing unit (CPU). The operating system is the one that is in charge of keeping track of where you are in each of these activities and allowing you to transition between them without losing any data in the process. It is responsible for keeping track of where you are in each of these tasks.

Despite the fact that early operating systems did not offer full support for multitasking, it was nonetheless feasible to run a large number of apps concurrently on those systems. Because of this, a single piece of software may use the full of the central processing unit (CPU) of the computer when it is working on a particular job. The user was unable to finish other activities, such as opening and closing windows, because essential operating system procedures, such as copying files, were taking up too much time. Other operations included opening and closing windows. The multitasking capabilities of today's operating systems, which are far more developed than those of operating systems in the past, allow for a large number of programs to be run concurrently without causing any interference with one another. In addition, several processes included inside the operating system are capable of running simultaneously.

1.9.1 Many Different Methods of Multitasking

There are primarily two different types of multitasking that one may perform. Each of these is explained in more detail below:

1. juggling many responsibilities while keeping an eye on safety
2. Working on many tasks simultaneously while collaborating with others and performing multiple jobs in advance

A computer's operating system is burdened with a special obligation known as preemptive multitasking. This responsibility is unique to the operating system. Before

going on to assigning another activity to use the operating system, it is its role to calculate the amount of time spent by a single task before moving on to the next activity. To give it its name, a 'preemptive' operating system is one in which the operating system itself controls the whole process.

The term "preemptive multitasking" refers to the method of multitasking that is utilized by desktop operating systems. Unix was the first operating system to use this method of multitasking, and as a result, it is considered the pioneer of this approach. Windows NT and Windows 95 were the first versions of Windows to have preemptive multitasking in their respective operating systems. With the debut of OS X, the Macintosh received the ability to multitask in a proactive manner. The apps will get a signal from this operating system letting them know when it is time for them to relinquish control of the CPU to another program.

1.9.2 Performing Multiple Tasks While Working Together

By the way, when individuals refer to "non-preemptive multitasking," they are referring to cooperative multitasking as the concept they have in mind. Cooperative multitasking has as its major goal the completion of the task that is now being worked on while simultaneously freeing up the central processing unit (CPU) for use in another activity. `taskYIELD` is the application that must be utilized in order to successfully complete this project. Whenever the `taskYIELD()` function is called, the context-switching process is carried out as part of that call.

Both Windows and MacOS had the capability of cooperative multitasking at their disposal. After a small piece of work in response to a message has been finished, a Windows application will then return control of the computer's central processing unit (CPU) to the operating system until the program receives another message. This procedure will continue until the program receives another message. It worked perfectly as long as every program was written carefully, without making any mistakes, and taking into account how it would interact with the other programs.

The advantages and disadvantages of being able to work on many projects at the same time. The following is a summary of several of the advantages and disadvantages that are connected with multitasking:

- **Additional Advantages:** The capacity to perform several tasks at once brings with it a variety of advantages, including the following ones:

Watch out for a Certain Amount of Customers. This operating system is more suited to supporting numerous users at the same time, and it enables a seamless functioning of various apps without significantly compromising the performance of the system as a whole. Additionally, this operating system is better suited to supporting several devices at the same time.

- **Virtual storage of one's memory:** Operating systems that are able to support a large number of processes frequently use the most modern technology available for their virtual memory. Because of virtual memory, a program does not have to rely on a considerable amount of waiting time in order to carry out its tasks. If an issue arises, the programs in question are moved to virtual memory.
- **Dependability of the Highest Order:** Operating systems that allow for multitasking give numerous users with additional flexibility, which in turn makes those users happy. It grants the ability for each user to execute a single application concurrently as well as several programs at the same time.
- **Memory that is Kept Safely Away:** Operating systems that support multitasking have procedures in place to keep memory management neat and orderly. Since this is the case, the operating system does not create any exceptions or provide any permissions to apps that ought not to be running in order to waste RAM.

1.10 TIME SHARING

An operating system is nothing more than a collection of software that manages the resources of the computer hardware and gives the apps that run on the computer access to common features. It is also usually referred to as an OS, which is an abbreviation for operating system. It is one of the numerous components that make up the software that operates the system on the computer, and it is also one of the most important components. This component is the operating system. There are many different kinds of operating systems, but one of the most essential types of operating system is called a time-sharing operating system. There are also many more forms of operating systems.

Because of the process of time-sharing, a specific computer system may be able to allow the concurrent use of its software by a sizeable number of users, each of whom may access the system from a separate terminal. This may be possible because the time-sharing technique is implemented on the system. When one has mastered the

technique of multiprogramming, the next natural step is to move on to the concept of multitasking, which is also frequently referred to as time-sharing systems. This is the next logical step. Utilizing a single processor to accommodate the needs of several users at the same time is an example of the practice known as time sharing.

Multi programmed Batch Systems and Time-Sharing: Systems are fundamentally distinct from one another due to the fact that although the goal of Multi programmed Batch Systems is to increase processor utilization, the aim of Time-Sharing Systems is to decrease reaction time. This is the primary distinction between the two. This is the major distinction that can be made between the two different kinds of systems. This is the most significant difference that can be made between the two different sorts of systems that are available.

The central processing unit (CPU) is able to do a great number of tasks simultaneously due to the fact that it may transition between the many activities that it is responsible for. Despite this, there is a significant amount of switching between tasks. because we want to ensure that the user receives a prompt response to the question that they have asked. If we take transaction processing as an example, the processor will run each user's application in a brief burst of computing, which is also referred to as a quantum of computation. This short burst of computing may also be thought of as a quantum of computation. This kind of computation is lightning fast and very effective. This suggests that in the case that there are n users present, a time quantum will be distributed to each and every one of those users individually. The longest amount of time that can elapse between the time the user sends in the command and the time when a response is delivered is a few seconds at most.

An operating system will allot a minute amount of time to each user by using CPU scheduling and multiprogramming to divide up the available time. In this way, the time may be used more efficiently. This time will be distributed evenly between all of the users. The architecture of various different computer systems, which were originally developed with the major focus on batch processing, has been modified so that they may now be utilized as time-sharing systems. This shift was brought about as a result of the fact that these computer systems are now more versatile. Operating systems that make use of timesharing offer a number of benefits, some of which are as follows:

- One of its most notable advantages is that it makes a rapid reaction possible.
- It lessens the amount of software that is duplicated.

- It lessens the amount of time that the central processing unit (CPU) of the computer is idle.

The use of time-sharing operating systems is fraught with a number of drawbacks, some of which are as follows:

- The dependability of time sharing is now one of the things that is giving us pause for concern.
- It appears that there is an issue with the transfer of the data.
- There is a chance that the confidentiality and security of user apps and data will be compromised.

1.11 PARALLEL

Parallel processing is a form of computer processing that allows for the execution of software programs to be completed more quickly than it would be otherwise possible. In order for this kind of system to function properly, a program must first be segmented into numerous bits before those pieces may be processed simultaneously. Tightly coupled systems are the sort of systems that may have more than one processor and go by the name of the same name. Parallel systems are those that make use of many computer resources at the same time. These systems can be implemented on a single computer with several processors, on a number of computers that are linked together over a network to form a parallel processing cluster, or on a combination of all of these different configurations. Utilizing several computers at the same time is another facet that might be a part of parallel processing systems.

Parallel computing is a natural step from serial computing in which workloads are sliced up into smaller, more manageable portions that are then executed in parallel with one another. In addition, there is a set of instructions supplied for every single component. The instructions for each component are processed by distinct central processing units (CPUs) in parallel with one another.

Programming parallel systems is more difficult than programming computers with a single processor because the architecture of parallel computers varies proportionally, and the operations of several CPUs must be coordinated and synchronized. This makes it more difficult to program parallel systems than it is to program computers with a

single processor. The programming of parallel systems is thereby made more difficult. There are a wide variety of various topologies that may be utilized to link processors and memory modules, and each of these topologies requires a different strategy when it comes to programming. The three types of processors that are employed the most commonly in the creation of parallel computers are synchronous processors, in which each one has its own memory; asynchronous processors, in which each one has its own memory; and asynchronous processors with a common, shared memory. Flynn has arranged the computer systems into groups according to the amount of parallelism that may be found in either the data streams or the instructions. These are the ones:

1. An abbreviation for "single stream of instructions and single stream of data," or "SISD" for short.
2. The SIMD architecture, often known as the Single Instruction Stream, Multiple Data Stream architecture.
3. Multiple Instruction Streams and a Single Data Stream is the abbreviation for MISD.
4. Multiple Instructions, Multiple Data, or MIMD, which stands for multiple data streams and multiple instruction streams.

The previous classification of parallel computing systems is based on two separate features, namely the number of data streams and the number of instruction streams that may be handled concurrently. These two numbers are known as the number of data streams and the number of instruction streams respectively. The word "instruction stream" refers to an algorithm that informs the computer what to do, whereas the term "data stream" refers to the data that are being operated upon and is identical with "input to an algorithm." In this particular context, the term "instruction stream" refers to an algorithm that teaches the computer what to do.

Only two of Flynn's categories may be used to describe parallel computers, despite the fact that he split computer systems into four groups according to their parallelism. The SIMD and MIMD versions of computers are the ones in concern here.

Each processing unit of a Single-Instruction Multiple-Data (SIMD) computer deals with a different item of data and gets a separate stream of instructions from a centralized control unit. The capacity of a SIMD computer to do several tasks

concurrently is one of its defining characteristics. The vast majority of SIMD computers run in synchronous mode and only include a single global docking station. The block diagram for a SIMD computer looks like this as an example:

1. Computers that use the MIMD architecture are constructed out of n independent processing units, each of which processes its own distinct stream of instructions and each of which operates on its own distinct piece of data. The Most Integrated Multiprocessor Device (MIMD) is the most powerful computer system that incorporates the whole range of multiprocessor systems. Its full name is the Most Integrated Multiprocessor Device. The MIMD computer's block diagram is presented for your perusal down below.
2. The programming of SIMD systems is substantially less complicated than other types of computer systems since they only use a single execution thread. On the other hand, the MIMD machines are more efficient since you are able to employ the full power of the machine at all times. This results in a greater overall output.
3. The management of the resources that are made accessible by parallel computers is the fundamental concern of the operating systems that are utilized by parallel computers. A parallel computer is a term that refers to a network of computers that are able to work together to solve a problem that arises in the field of computing. Therefore, a parallel computer may be a supercomputer with hundreds or thousands of processors, or it could be a network of workstations networked together in a single system.
4. Back in the day, you could only find parallel computers in research laboratories, and their major function was to complete computationally difficult tasks such as numerical simulations of complex systems. Parallel computers are still used for these purposes today. There is a significant selection of parallel computers offered for sale on the market at the present time. In the commercial sector, these computers are used to carry out operations that are heavy on data, but in the scientific and engineering sectors, these computers are used to carry out projects that are heavy on calculations.
5. The fact that new applications are continually being developed in today's world makes it necessary to make use of more powerful computers. The vast majority of applications being executed on parallel computers are business-related in some way. A computer system that is capable of executing such an application should have the ability to do complicated data processing on huge volumes of

data. This area includes a wide variety of applications, some of which are as diverse as graphics and virtual reality, as well as decision help, parallel database administration, and medical diagnostics. Commercial applications will, without a doubt, be the ones to determine the architecture of future parallel computers, although research applications will continue to be the primary consumers of parallel computing technology. This fact cannot be refuted under any circumstances.

Concurrency soon emerges as an essential requirement when it comes to the development of algorithms and systems. Not only does a program need to be able to operate on a computer architecture that is capable of running many processors, but it also has to be able to utilize whatever number of processors that are available. Tanenbaum describes a distributed system as a group of self-sufficient computers that give the user the appearance that they are functioning as a single unit. Because of this, the computers themselves need to be able to function independently, and the software should be able to hide the fact that there are many machines from the users. The workstations and MIMD computers that are connected to each other over LAN and WAN are two examples of distributed systems. These systems are all connected to each other. The major difference between parallel and distributed computer architectures is the method in which these two separate kinds of computer systems are put to use in a computing environment. In a parallel system, many processing units can work together to address a single problem at the same time. A system that allows several people to use it at the same time is referred to as a distributed system.

1.12 DISTRIBUTED & REAL-TIME O.S.

A distributed operating system (DOS) is an essential type of operating system. Distributed systems use many central processors to serve multiple real-time applications and users. As a result, data processing jobs are distributed between the processors. It connects multiple computers via a single communication channel. Furthermore, each of these systems has its own processor and memory. Additionally, these CPUs communicate via high-speed buses or telephone lines. Individual systems that communicate via a single channel are regarded as a single entity. They're also known as loosely coupled systems.

This operating system consists of numerous computers, nodes, and sites joined together via LAN/WAN lines. It enables the distribution of full systems on a couple of

center processors, and it supports many real-time products and different users. Distributed operating systems can share their computing resources and I/O files while providing users with virtual machine abstraction.

1.12.1 Real-Time operating system

A real-time operating system, often known as an RTOS, is a particular kind of computer operating system that is used in computers and places severe time constraints on the completion of any job. This sort of operating system is frequently referred to as an RTOS. It is applied the most frequently in those systems in which the results of the calculations are used to influence a process while it is still in the process of being carried out. With the aid of a sensor that is monitoring the event, the information about the event is transmitted to the computer whenever anything occurs that is unrelated to the computer. The operating system understands the signal that the sensor sends out as an interrupt since it is the signal that the sensor itself generates. After the operating system has been presented with an interrupt, it will initiate a specific process or a collection of processes in order to deal with the interrupt.

This procedure will not be halted in any manner while it is being carried out; the only exception to this rule will be if something with a higher priority causes an interruption. As a consequence of this, the various interrupts ought to adhere to a tight priority ordering that has been established. The interruption that has the highest priority should be allowed to initiate the process, while interruptions with lower priorities should be saved in a buffer and dealt with at a later time if necessary. In an operating system like this one, one of the most important functions is the management of interrupts.

In real-time computing environments, general-purpose operating systems are unable to achieve the degree of performance that is necessary, thus instead, specialized operating systems that are designed for a specific purpose are used.

CHAPTER 2

PROCESS AND THREADS MANAGEMENT

2.1 PROCESS CONCEPT

One way to think of a process is as a plain application running on a computer. The term "process" refers to a function of the operating system that is being performed in the background by an application that is now active. The Process is the most important aspect of any and all actions that are associated with computers. The method is not the same as using computer code, despite the fact that the practice is quite similar to computer code. A process, on the other hand, is a "active" entity in its own right, in contrast to the program, which is generally thought of as a "passive" entity in and of itself. Among the qualities that are held by the process are a wide variety of additional characteristics, such as the present status of the hardware, the quantity of RAM, and the CPU.

2.1.1 A process that is carried out within the context of an operating system

A process is the act of actively running software or computer code. Processes can be either manual or automated. Any procedure must be carried out in the precise order in which it was described. A process is an item that helps in representing the fundamental work unit that has to be implemented in any system. Processes may be thought of as procedures. This essential component of the work that needs to be done must be completed in order for the system to perform as intended. Computer programs, in other words, are created as text files, and when they are executed, they produce processes that carry out all of the tasks that are outlined in the program. These processes are initiated once the actual software that was downloaded is run.

When a program is loaded into memory, it may be partitioned into a process by being broken down into its four component pieces, which are the stack, the heap, the text, and the data. This allows the program to run more efficiently. The operation that takes place in the main memory is depicted in a more condensed form in the following image.

- **Replaced with:** The return address, local variables, and arguments sent to methods and functions are all saved on the process stack. This is also the location where information that is only momentarily required may be found.

- **Heap:** This is the portion of memory that is dynamically allotted to a process while that process is running in the system. Text This contains the data that has been saved in the registers of the processor as well as the most recent action, as represented by the value of the program counter. Also included is the data that has been saved in the registers of the memory.
- **Data:** In this section of the essay, we are going to discuss global variables in addition to static variables.
- **It is the program:** A program is a set of instructions that are carried out whenever a specific job is given permission to finish carrying out that specific duty. These instructions are carried out whenever a program is executed. The applications are often written in a programming language such as C, C++, Python, Java, R, C # (C sharp), or another language that is very close to C # (C sharp).

A set of instructions that, when carried out by a computer, make it possible for the work at hand to be finished effectively is what we refer to as a computer program.

Difference between process and the program:

S. No	Process	Program
1	A process is actively running software or a computer code. Any procedure must be carried out in a precise order. An entity that helps in describing the fundamental work unit that must be implemented in any system is referred to as a process	Program is a set of instructions which are executed when the certain task is allowed to complete that certain task
2	Process is Dynamic in Nature	Program is Static in Nature
3	Process is an Active in Nature	Program is Passive in Nature
4	Process is created during the execution and it is loaded directly into the main memory	Program is already existed in the memory and it is

		present in the secondary memory.
5	Process has its own control system known as Process Control Block	Program does not have any control system. It is just called when specified and it executes the whole program when called
6	Process changes from time to time by itself	Program cannot be changed on its own. It must be changed by the programmer.
7	A process needs extra data in addition to the program data needed for management and execution.	Program is basically divided into two parts. One is Code part and the other part is data part.
8	Processes have significant resource demands; they require resources like Memory Addresses, Central Processing Unit, Input or Output until their presence or existence in the Operating System.	A program just needs memory space to store its instructions; no further resources are needed.

2.1.2 Process Control Block

An operating system may be able to aid in the process of establishing new processes, scheduling them, and terminating them with the assistance of a Process Control Block. The Process Control Block (PCB), which is a part of the Operating System, is one of the components that helps with the control of the processes that are involved in functioning. Each and every one of the operating system's processes has a corresponding Process Control Block associated with it. Because it saves data about a number of things, including the state of the processes, the condition of their I/O, and the scheduling of their CPU, a PCB is able to keep track of the processes that are running on it.

Now, let's try to obtain a better knowledge of the Process Control Block with the assistance of the components that are already a part of the Process Control Block. An example of a process control block has the following components:

1. Procedure Identifier
2. The Process and its Present State
3. The Refutation of the Program
4. The computer's registries (registers)
5. Specifics Regarding the Programming of the Central Processing Unit
6. Data and Record-Keeping Concerning Companies and Their Operations
7. Information Regarding the Administration of Memory
8. Information Regarding the Status of the Inputs and Outputs

Now that we've covered the basics, let's go into the specifics of each and every one of these parts.

1. **The ID of the Procedure:** It acts as a distinguishing sign for the procedure that is being carried out at the moment. Discovering the procedure is made much easier with the assistance of this useful tool. A further use for it would be to recognize the process itself, which is where it demonstrates its value in a significant way.
2. **The Existing Context of the Process:** Now, please educate us on all there is to learn about each and every one of the stages of the procedure. Please allow me to go into further detail about each and every one of the states.
 - **The State of New Hampshire:** The state of having a program that will be picked up by the operating system and placed directly into the main memory is known as being in the New Process State.
 - **Being in a state of readiness:** A process instantly enters the ready state when it has been formed. While in this state, it waits for the central processing unit (CPU) to be assigned to it. The operating system will recover newly formed

processes from secondary memory and relocate them into main memory, where they will remain. Secondary memory is used just temporarily whereas main memory is used permanently. The processes that are now saved in the main memory and are prepared to be executed are said to be in a "ready state," and the term "ready state processes" is used to characterize those processes. It is possible that a number of separate processes are occurring simultaneously at this time.

- **A Condition of Constant Motion:** The Operating System will select one of the processes that is in the ready state for further processing in a manner that is in compliance with the scheduling system. If our computer system just has one central processing unit, often known as a CPU, then there will never be more than one process running at any given moment. If the computer has n processors, then we will be able to execute n separate processes at the same time regardless of which one we choose to do.
 - **A Position Used for Holding Up Traffic or Waiting:** Depending on the scheduling system that is in place or the behavior that is inherent to the process itself, a process can transition from the Running state to either the Block state or the Wait state after it has been in the Running state for some amount of time. The operating system will place a process in a blocked or waiting state and will reassign the CPU to other processes as it waits for a certain resource to be allocated or for input from the user. as it waits for these events, the operating system will reassign the CPU to other processes.
 - **The State of Having Completed or Finished Something:** When a process has completed all of its execution, it will move into the termination state and remain there until it is stopped. The operating system will abort the process and remove the whole context of the process, which is referred to as the Process Control Block.
3. **A Countdown Timer for Applications:** A register called a program counter (PC) is located within the central processor unit of the computer. This register is responsible for storing the location of the next instruction that the computer will read from memory in order to carry out. A digital counter is required for both monitoring the present stage that a task has reached in its execution as well as assessing the speed at which jobs are being finished. One can also refer to a program counter by its various names, which include an instruction pointer,

instruction addresses register, sequence control register, and instruction counter. Another term for a program counter is a sequence control register.

- 4. The Computer's Onboard Registers:** When a process is actively running, the data that is now being processed by the registers of the central processing unit of the computer is saved in this place. This occurs whenever the process is operating. Accumulators, index and general-purpose registers, instruction registers, and condition code registers are some of the registers that may be found in a central processing unit (CPU). Other types of registers include condition code registers.
- 5. Specifics Regarding the Programming of the Central Processing Unit:** In order to successfully carry out an operation, it is necessary to make the necessary preparations. This timeline will establish the amount of time necessary for it to transition from the ready state to the functioning state. In addition to the process priority and scheduling queue pointers, which indicate the order in which instructions are to be processed, the CPU scheduling information includes a number of distinct scheduling parameters. These parameters determine the order in which instructions are to be performed.
- 6. Information Regarding:** Accounting and Business Included in the report on the State of Business Addressing and Information are facts such as the amount of actual time that a process takes up, the number of tasks or processes, and other information of a similar nature.
- 7. Information Regarding the Administration of Memory:** Memory Management Information is a section of the file that contains data pertaining to the page and segment tables, as well as the value of the base and limit registers. This section of the file may be accessed by [clicking here](#). The memory system of the operating system is essential to its functioning properly.
- 8. Information Regarding the Status of Input and Output:** This section of the input output status information comprises information regarding the status of input and output, such as specifics on the process statuses, amongst other things.

2.2 PROCESS STATES

A process has several stages that it passes through from beginning to end. There must be a minimum of five states. Even though during execution, the process could be in

one of these states, the names of the states are not standardized. Each process goes through several stages throughout its life cycle.

2.2.1 Process States in Operating System

The states of a process are as follows:

- **New (Create):** In this step, the process is about to be created but not yet created. It is the program that is present in secondary memory that will be picked up by OS to create the process.
- **Ready:** New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue called ready queue for ready processes.
- **Run:** The process is chosen from the ready queue by the CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- **Blocked or Wait:** Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or waits for the state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
- **Terminated or Completed:** Process is killed as well as PCB is deleted. The resources allocated to the process will be released or deallocated.
- **Suspend Ready:** Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought onto the main memory.
- **Suspend wait or suspend blocked:** Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused

them to move to secondary memory. When work is finished it may go to suspend ready.

- **CPU and I/O Bound Processes:** If the process is intensive in terms of CPU operations, then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.

How does a process move between different states in an operating system? A process can move between different states in an operating system based on its execution status and resource availability.

Here are some examples of how a process can move between different states:

- **New to ready:** When a process is created, it is in a new state. It moves to the ready state when the operating system has allocated resources to it and it is ready to be executed.
- **Ready to running:** When the CPU becomes available, the operating system selects a process from the ready queue depending on various scheduling algorithms and moves it to the running state.
- **Running to blocked:** When a process needs to wait for an event to occur (I/O operation or system call), it moves to the blocked state. For example, if a process needs to wait for user input, it moves to the blocked state until the user provides the input.
- **Running to ready:** When a running process is preempted by the operating system, it moves to the ready state. For example, if a higher-priority process becomes ready, the operating system may preempt the running process and move it to the ready state.
- **Blocked to ready:** When the event a blocked process was waiting for occurs, the process moves to the ready state. For example, if a process was waiting for user input and the input is provided, it moves to the ready state.
- **Running to terminated:** When a process completes its execution or is terminated by the operating system, it moves to the terminated state.

2.2.2 Types of Schedulers

1. **Long-term – performance:** Decides how many processes should be made to stay in the ready state. This decides the degree of multiprogramming. Once a decision is taken it lasts for a long time which also indicates that it runs infrequently. Hence it is called a long-term scheduler.
2. **Short-term – Context switching time:** Short-term scheduler will decide which process is to be executed next and then it will call the dispatcher. A dispatcher is a software that moves the process from ready to run and vice versa. In other words, it is context switching. It runs frequently. Short-term scheduler is also called CPU scheduler.
3. **Medium-term – Swapping time:** Suspension decision is taken by the medium-term scheduler. The medium-term scheduler is used for swapping which is moving the process from main memory to secondary and vice versa. The swapping is done to reduce degree of multiprogramming.

2.2.3 Multiprogramming

We have many processes ready to run. There are two types of multiprogramming:

1. **Preemption** – Process is forcefully removed from CPU. Pre-emption is also called time sharing or multitasking.
2. **Non-preemption** – Processes are not removed until they complete the execution. Once control is given to the CPU for a process execution, till the CPU releases the control by itself, control cannot be taken back forcibly from the CPU.

2.2.4 Degree of Multiprogramming

The number of processes that can reside in the ready state at maximum decides the degree of multiprogramming, e.g., if the degree of programming = 100, this means 100 processes can reside in the ready state at maximum.

2.2.5 Operation on the Process

1. **Creation:** The process will be ready once it has been created, enter the ready queue (main memory), and be prepared for execution.

2. **Planning:** The operating system picks one process to begin executing from among the numerous processes that are currently in the ready queue. Scheduling is the process of choosing the next process to run.
3. **Application:** The processor begins running the process as soon as it is scheduled to run. During execution, a process may become blocked or wait, at which point the processor switches to executing the other processes.
4. **Killing or Deletion:** The OS will terminate the process once its purpose has been fulfilled. The process's context will be over there.
5. **Blocking:** When a process is waiting for an event or resource, it is blocked. The operating system will place it in a blocked state, and it will not be able to execute until the event or resource becomes available.
6. **Resumption:** When the event or resource that caused a process to block becomes available, the process is removed from the blocked state and added back to the ready queue.
7. **Context Switching:** When the operating system switches from executing one process to another, it must save the current process's context and load the context of the next process to execute. This is known as context switching.
8. **Inter-Process Communication:** Processes may need to communicate with each other to share data or coordinate actions. The operating system provides mechanisms for inter-process communication, such as shared memory, message passing, and synchronization primitives.
9. **Process Synchronization:** Multiple processes may need to access a shared resource or critical section of code simultaneously. The operating system provides synchronization mechanisms to ensure that only one process can access the resource or critical section at a time.
10. **Process States:** Processes may be in one of several states, including ready, running, waiting, and terminated. The operating system manages the process states and transitions between them.

2.2.6 Features of the Process State

- A process can move from the running state to the waiting state if it needs to wait for a resource to become available.

- A process can move from the waiting state to the ready state when the resource it was waiting for becomes available.
- A process can move from the ready state to the running state when it is selected by the operating system for execution.
- The scheduling algorithm used by the operating system determines which process is selected to execute from the ready state.
- The operating system may also move a process from the running state to the ready state to allow other processes to execute.
- A process can move from the running state to the terminated state when it completes its execution.
- A process can move from the waiting state directly to the terminated state if it is aborted or killed by the operating system or another process.
- A process can go through ready, running and waiting state any number of times in its lifecycle but new and terminated happens only once.
- The process state includes information about the program counter, CPU registers, memory allocation, and other resources used by the process.
- The operating system maintains a process control block (PCB) for each process, which contains information about the process state, priority, scheduling information, and other process-related data.
- The process state diagram is used to represent the transitions between different states of a process and is an essential concept in process management in operating systems.

2.3 UNI PROCESSOR SCHEDULING

Within the confines of this inquiry, we are going to have a conversation about a variety of short-term scheduling techniques that are used in uniprocessor systems in a broad variety of different applications. After finishing this course of study, you should be able to give proper responses to the following inquiries: In what precisely does the process state consist? Just what does one mean when they talk about "short-term

scheduling"? When is the best time to execute scheduling for the time that is coming up soon? What additional kinds of scheduling algorithms do we have at our disposal at the moment? How do we go about choosing an algorithm for the schedule, and what elements do we base our selection on? The following table gives an overview of which aspects of this inquiry are dedicated to resolving the numerous issues that this study has brought to light.

2.3.1 The Operation of State Machines

- **Something completely novel:** the establishment of a protocol. The main memory receives a copy of the program's source code that was read from the disk.
- **Ready:** The process is idly waiting for a processor to be allocated to it while doing nothing productive in the meanwhile. The code for the software is kept in the main memory of the computer.
- **Running:** The instructions are now being executed by the central processing unit. The instructions themselves are stored within the CPU.
- **Waiting:** The process is halted because it is waiting for the occurrence of particular events (a signal). This causes the process to be referred to as "waiting." Either the main memory or the computer's virtual memory houses the instructions for the computer. The instructions have not been relocated to any other memory.
- **Finished:** This status denotes that the operation has finished being carried out in its entirety.

2.3.2 Several Alternative Routing Procedures

There are three main subcategories that can be found under the overarching category known as "scheduling." These subcategories include long-term scheduling, medium-term scheduling, and short-term scheduling. The organization of activities for the very near future is the major focus of this study's attention. This table provides an overview of when each of the three activities related to scheduling should be finished in respect to state changes.

The Scheduling Process and the Changeover from One State to Another

- Making Plans for the Not-So-Distant Future

- **Considerations Made Before Scheduling**

When determining the algorithm to use for scheduling, it is necessary to take into account a wide range of scheduling criteria. These elements may be broken down into several categories.

- Turnaround time in seconds per process (u%)
- CPU utilization as a percentage (u%)
- Amount of time spent waiting (in seconds)
- The response time, measured in seconds
- Throughput, measured in processes performed per second
- Limit on the amount of time you have to make your decision
- Fairness
- Predictability
- Enforcement of regulations
- Balance

2.3.3 Various standards for each of the available systems

Because each system has its own specific requirements, it is necessary to adhere to a unique set of criteria for each of those systems. Various Standards for the Many Different Types of System Algorithms Used in the Process of Scheduling

After that, we will proceed to discuss four distinct approaches to time management, each accompanied by multiple illustrations. In the next paragraph, we shall delve even deeper into detail in reference to these algorithmic techniques.

1. First-Come-First-Serve, sometimes referred to as FCFS (also shortened):

Decisions reached Pick the process that will get you to the end result in the shortest period of time. (This is not a case of self-defense)

The following is an illustration of an FCFS: Let's imagine there are four distinct processes, and the table comprises the times at which they will arrive as well as the times at which they will provide their services.

A Table of the Processes Involved in FCFS: The following is what transpired as a result of the scheduling done on the Gantt chart:



Fig 2.1

2. Round-Robin (RR):

It is the responsibility of the selection function to guarantee that each process completes its execution for the allotted amount of time quantum. (In advance of the schedule)

The following is an illustration of RR: Let's imagine there are four distinct processes, and the table comprises the times at which they will arrive as well as the times at which they will provide their services. Round-Robin is a table of the processes that include rotation.

The following is what transpired as a result of the scheduling done on the Gantt chart:

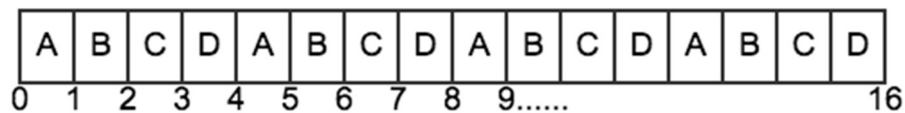


Fig. 2.2

3. Shortest-Job-First (SJF):

The following components make up the selecting function: Select the non-preemptive strategy that calls for the fewest number of service hours to be performed.

An illustration for the SJF is as follows: Let's imagine there are four distinct processes, and the table comprises the times at which they will arrive as well as the times at which they will provide their services.

A Table of Procedures for the SJF: The following is what transpired as a result of the scheduling done on the Gantt chart:

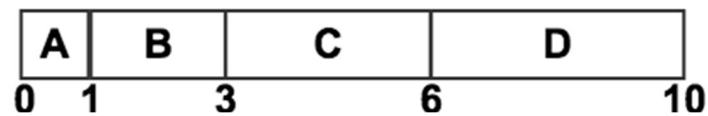


Fig. 2.3

SJF Scheduling Result

2.4 TYPES OF SCHEDULING

2.4.1 Putting Forward Efforts for the Long Term

When a new method is established, it is standard practice to carry out some kind of long-term planning strategy. You may see an example of it in the next paragraph. If the number of ready processes in the ready queue were to significantly increase, then the amount of processing time spent by the operating system (i.e., the processor) on activities such as maintaining long lists, switching contexts, and dispatching would also increase. Because of this, there should be a cap placed on the number of processes that are permitted to join the ready queue. The individual referred to as the "long-term scheduler" is the one responsible for handling this.

The long-term scheduler is in charge of deciding which programs may enter the system to be processed, and it is their job to choose which ones can do so. When a task or process is accepted for the very first time, it instantly transforms into a process and is added to the queue for the short-term scheduler. In some computer systems, a newly produced process will begin in a swapped-out situation rather than in its normal state. If this is the case, the process is added to a queue for the medium-term scheduler, which is responsible for managing queues in order to minimize the amount of time that is spent waiting in line and maximize the performance of the system as a whole.

The long-term scheduler is responsible for limiting the number of processes that may be handled. It accomplishes this by deciding whether or not to add one or more new tasks depending on criteria such as priority, execution time, input/output requirements, or FCFS (First-Come, first-serve), which stands for "First-Come, first-serve." The overall number of procedures that can be processed is cut down as a result of this. The

long-term scheduler is only going to operate intermittently the vast majority of the time.

2.4.2 Making plans for the immediate and not too distant future

The medium-term scheduling component is one of the sub-functions that are included in the swapping function. When a portion of the main memory becomes available once more, the operating system performs a scan of the list of processes that are suspend-ready and decides which one of those processes is going to be swapped in (taking into consideration a variety of factors including priority, the amount of memory and other resources that are required, etc.). This scheduler collaborates extremely closely with the one responsible for long-term planning in order to guarantee that everything goes well. It will be responsible for switching in among the processes that are being switched out in order to carry out its task. The scheduler for the medium term is utilized somewhat more frequently than the scheduler for the short term.

2.4.3 In the immediate future The schedule itself

When referring to the short-term scheduler, the term "dispatcher" can also be used. The short-term scheduler is activated anytime an event occurs that has the potential to cause the process that is presently being executed to be terminated. This means that the short-term scheduler is called into action whenever such an event takes place. Instances including but not limited to clock interrupts, input/output interrupts, requests to the operating system, signals, and so on. The short-term scheduler is the one that is used most frequently in this system. It decides which of the processes that are ready to execute should receive the processing power of the computer and then selects one of the processes from the available options. It frequently has to decide on a new procedure for the central processing unit to carry out. It needs to move at a lightning-fast pace.

2.4.4 Considerations Made Before Scheduling

Both "scheduling approach" and "scheduling criteria" refer to the same thing in their respective contexts. When it comes to multitasking, scheduling is one of the most crucial elements to consider. The many different algorithms for scheduling CPU time each have their own individual set of characteristics. The following is a selection of the factors that were considered in making this comparison between the various algorithms:

- **Utilization of the CPU:** It is imperative that the central processing unit be kept as active as is humanly possible. There is a range of possibilities for the value, from 0% to 100%. In point of fact, estimates vary widely, ranging anywhere from forty to ninety percent.
- **Throughput:** The term "throughput rate" refers to the rate at which tasks are completed in a certain period of time. This pace is referred to as the throughput rate.
- The amount of time required to carry out a process in its entirety, from beginning to end, is referred to as the turnaround time. To ascertain it, simply count the number of seconds that have elapsed since the commencement of the process until it was brought to a successful finish.
- Waiting time is the entire amount of time that has been spent waiting in the ready line. The term "waiting time" refers to this whole length of time.
- **Response time:** the amount of time that elapses between the moment that the submission was made and the time that a response is initiated is referred to as the response time. The amount of time that elapses between the point in time when a request is submitted to the point in time when the very first answer is created is what serves as the basis for calculating it.
- Take measures to ensure that all running processes receive an equitable share of the available CPU resources.

Scheduling using Non-Preemptive Availability: In non-preemptive mode, once a process enters the running state, it will continue to execute until it either terminates itself or blocks itself to wait for input/output or by contacting some operating system service for assistance. Preemptive mode allows processes to stop themselves from executing after they have entered the running state. This happens regardless of whether the process is operating in the preemptive mode or the non-preemptive mode.

Preemptive Scheduling: When employing preemptive mode, the operating system may choose to interrupt a process that is currently running and shift it to the ready state. This is an example of the preemptive scheduling technique. The use of this strategy is referred to as "preemptive scheduling."

It is feasible that the non-preemptive version of a policy will have less overhead than the preemptive version of the policy; yet, the preemptive version of the policy may offer better service if a new process is introduced or whenever there is an interruption. It is preferred to have the maximum possible CPU use and throughput, as well as the lowest possible reaction time, waiting time, and turnaround time. This is because it allows for the most efficient use of resources.

2.5 SCHEDULING ALGORITHMS

The majority of the time, the management of short-term scheduling is accomplished with the assistance of scheduling algorithms or scheduling laws. The basic goal of short-term scheduling is to allot processor time in such a manner as to improve the performance of some component or features of the system's overall behavior. This is accomplished by allocating the time in such a way as to maximize efficiency. When utilizing these scheduling algorithms, the assumption is made that there is just a single processor that is accessible. Scheduling algorithms choose which of the processes in the ready queue are to be given to the CPU depending on the kind of scheduling policy and whether or not that policy is preemptive. This selection is made based on the kind of scheduling policy and whether or not that policy is preemptive. When compared to preemptive scheduling strategies, non-preemptive scheduling policies allow for a greater number of processes to execute in parallel. When it comes to making the schedule, both the arrival time and the amount of time that will be spent providing the service will be taken into consideration. The following is a list of the scheduling algorithms that can be used:

- Round-Robin Scheduling algorithm
- Non-preemptive priority Scheduling algorithm
- Preemptive priority Scheduling algorithm
- Round-Robin Scheduling algorithm
- Highest Response Ratio Next (HRRN) algorithm
- Multilevel Feedback Queue Scheduling algorithm
- Multilevel Queue Scheduling algorithm

- Multilevel Multilevel Queue Scheduling algorithm
- First-come, first-served scheduling (FCFS) algorithm
- Shortest Job First Scheduling (SJF)

The information that is provided below, which is laid out in the following way, is what we would use to describe the different scheduling policies that are at your disposal: Process, arrival time, and service time (sometimes referred to as burst time) are the three factors that go into determining priority. The scheduling is based on the "first come, first served" or "FCFS" concept, which is written as P1 | 0 | 4 | 2, P2 | 3 | 6 | 1, P3 | 5 | 3 | 3, and P4 | 8 | 2 | 1.

In the first place When they are gone, they are gone. When it comes to scheduling, the "first item in, first item out" method is utilized. When a process has completed all of its steps and is prepared, it is added to a queue that is referred to as "ready." When the process that is presently being carried out reaches a point where it is unable to complete its duties, the next process in the Ready queue that is selected for execution is the one that has been there the longest. This procedure was the first one to be entered into the ready queue, which contains all of the processes that are now available to be executed. The average amount of time spent in line at FCFS is often quite a significant period of time. This is not a preventative measure in any way.

The total of the waiting time and the amount of time spent providing service is the TURNAROUND TIME.

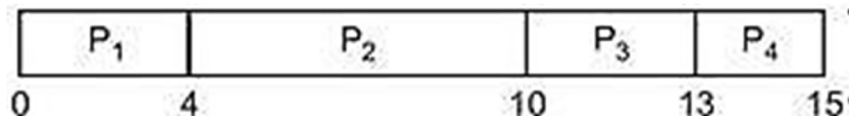


Fig. 2.4

SJF Scheduling, which stands for "shortest job first," is a method of organizing work shifts.

Using this strategy, the length of the next burst of activity by the CPU is distributed evenly across all of the separate processes. Shortest Job First, or SJF, is a method of scheduling that is also known as Shortest Process Next, or SPN. The activity that will

actually be carried out is determined to be the one that will require the fewest resources and the shortest amount of time to accomplish out of all of the operations that are prepared to be carried out. As a result, a process that can be completed in a short length of time will be given more priority than activities that require a large amount of time to complete. When the consecutive CPU bursts of two different processes are the same, FCFS scheduling will be used to break the tie between the processes. It is quite likely that the SJF scheduling algorithm is the most effective one.

It calculates the least feasible average time for a certain set of procedures and gives the results. It is not possible to implement it at the level of short-term scheduling for the CPU since it would cause too much overhead. There is no way to find out how long the shortest CPU burst that is feasible is going to be. Applications of SJF can be done in either a preemptive or non-preemptive manner; both are feasible. A preemptive SJF algorithm will bypass the process that is now being carried out if there is a chance that the subsequent CPU burst of a newly arriving process will be shorter than what is still available for the process that is currently being carried out.

The procedure that is now being carried out will be able to reach a fruitful and satisfying finish with the assistance of a Non-preemptive SJF algorithm. There are occasions when the Preemptive SJF Scheduling technique is sometimes referred to as the Shortest Remaining Time First algorithm. This is because of how the approach works.

CHAPTER 3

CONCURRENCY

3.1 PRINCIPLES OF CONCURRENCY

When we talk about concurrency, we are referring to the simultaneous execution of several sequences of instructions by multiple threads. This problem arises for the operating system if there are a significant number of process threads that are working in parallel. The process threads that are now running connect with one another on an ongoing basis, either by using shared memory or by-passing messages back and forth. When resources are shared in a concurrent environment, it can result in problems such as deadlocks and a reduction in the number of resources that are accessible.

It is useful in techniques that try to increase throughput, such as coordinating the execution of processes, allocating memory, and scheduling execution, all of which may be done using this information. Allowing for concurrent execution can be beneficial for a variety of reasons, some of which are as follows:

- The equitable distribution of material resources: Due to the limited availability of hardware resources, this environment supports several users at once.
- Sharing logical resources necessitates reusing the same file (or piece of information) for several different applications.
- The quickening of computational processes by the application of parallel processing
- Modularity entails breaking down the many system functions into their individual processes so that they may be managed independently.

3.1.1 The Relationship Between a Number of Different Processes

The following types of processes can be executed inside the confines of an operating system, and they can be grouped together as described in the following table:

- Methods and Procedures That Are Completely Unrelated

- Different Approaches to Working Together

Procedures That Are Completely Independent: Its status is never sent to any of the other processes that are currently being executed in the system.

- The only thing that will determine the result of the execution is the state that will be sent to it as input.
- The results of the execution will not change no matter what data is entered into the system since the results are always the same.
- The outcome of the independent procedure will have absolutely no impact on the other proceedings in any way, shape, or form.
- A system is said to be cooperating when the state of the system is conveyed to other processes inside the system.
- The result of the execution is non-deterministic, which means that it is not possible to predict it in advance and that it is dependent on the relative order in which the stages of the execution are carried out.
- There is no assurance that the result of the execution will always be the same in response to the same input. This is because there is no mechanism to ensure this.
- If the process of working with one another is stopped, it is possible that it will have an influence on other processes.

3.1.2 Procedures that are Carried Out During Operations

The majority of systems include support for at least two separate kinds of operations, each of which may be executed on either the creation or deletion of a process. This is possible because most systems may create or delete processes in either direction.

The development of a process begins with the creation of a parent process, which is followed by the construction of any processes that derive from that parent process. When there is more than one process established, there are several possible implementations that might be employed.

Running is something that may be done simultaneously by a parent and a child. The Parents have decided to put off ending their relationship until all of their children have gone through with the same decision.

- The parent and their children share the available resources in an equitable manner among themselves.
- Neither the father nor the children have anything in common with regard to their finances.
- The children are only given access to a fraction of the resources that are owned by their parents.

The successful conclusion of the procedure: The following is a list of some of the methods that may be used to terminate a child process:

- For any one of the following circumstances, a parent has the authority to stop the execution of one of their children:
 1. The child's level of resource consumption has reached the maximum level permitted for their allocation.
 2. The young child is relieved of the need to fulfill the task that had been assigned to them.
- If a parent has ended their connection with the kid, it is mandatory that the child's parental rights be terminated as well.

Although it is feasible to consider both interleaved and overlapping processes to be examples of concurrent processes, both of these types of processes have the same difficulties. It is not feasible to offer a prediction that is correct regarding the relative speed of the execution. It depends on the following factors:

- The activities that are carried out by various other processes
- The guidelines that the operating system establishes for scheduling and for interruptions
- The ways in which the operating system handles disturbances and disruptions

The following is a list of potential problems that might occur as a result of employing concurrency:

- Facilitating the secure distribution of global resources can be a tough endeavor. When two processes both make use of the same global variable and carry out read and write operations on that variable, the order in which the various read and write operations are carried out becomes an incredibly significant consideration.
- The challenge of getting the operating system to handle its resources in the best possible way The operating system faces a number of challenges in order to accomplish effective management of the resources.
- **Locating programming errors** — It is quite challenging to locate a programming issue since reports are often impossible to be replicated. This makes the process very tough. Because of this, the assignment is going to be quite difficult.
- **Locking the channel** - It is possible for the operating system to be inefficient if it only locks the channel and stops other processes from using it. This is because locking the channel prohibits other processes from utilizing the channel.

Concurrency affords its users a variety of advantages, some of which are enumerated in the following paragraphs:

- **The ability to run several programs concurrently** - This feature enables users to run many applications at the same time.
- A more efficient use of resources since it is now feasible to repurpose resources that are being left unused by one program and put them to use in another application. This results in a greater overall utilization of available resources.
- An improvement in the typical amount of time required to respond – In the absence of concurrency, each application would have to be run to completion before the next one could be run. This results in much longer response times.
- Improved performance as a direct effect of its usage in the operating system It enables the operating system to have improved performance as a direct result

of its use. The amount of time necessary to run both programs concurrently until they are done will be less than the amount of time that is required to run each application in sequential order. When one application utilizes only the central processing unit (CPU), and another application uses just the disk drive, the amount of time that is required to run both applications concurrently until they are finished will be less.

Concurrency can have a number of unfavorable effects, including the following:

- It is absolutely necessary to protect many programs from one another.
- It is important to coordinate the use of many different applications utilizing a variety of different approaches.
- Additional performance overheads and complexity in operating systems are required in order to move between different applications.
- There are some circumstances in which the execution of an excessive number of programs in parallel will result in a considerable loss in performance.

Problems Associated with Concurrent Activity:

- **Non-atomic:** Operations that are not atomic and might be stopped by numerous processes could be troublesome and lead to mistakes. These kinds of interruptions can occur because of the non-atomic nature of these operations.
- **Race conditions** - A race condition is when the outcome is contingent on which of several processes reaches a point first. This might happen when there are a lot of different processes going on at the same time. This is something that is possible when there are numerous processes going on at once.
- **Blocking:** If a process is waiting for a resource, it runs the risk of becoming blocked while it does so. When a process is waiting for input from a terminal, it is possible for the process to become stuck for a considerable length of time and remain in this state. This would be a highly undesirable consequence if the operation requires particular data to be updated on a frequent basis. If this is the case, then the procedure should be avoided.
- The process suffers from starvation when it does not obtain the essential service in order to advance.

- A deadlock is a circumstance that occurs when two processes become halted and, as a result, none of them can go forward to execute.

3.2 SEMAPHORES

In the discipline of computer science, a variable or an abstract data type is known as a semaphore. Its goal is to limit access to a common resource by several threads and to prevent critical section difficulties in concurrent systems such as multitasking operating systems. These issues might arise when there is insufficient access to the resource. Semaphores are an example of a primitive kind of synchronization that has been used. An example of a fundamental semaphore would be a simple variable that is changed in response to conditions that have been established by the programmer. This variable's value can be changed in a number of different ways, including being incremented, decremented, or toggled.

One helpful way to conceive of a semaphore as it is employed in real-world systems is as a record of how many units of a certain resource are available. This is one use of the semaphore. This record is connected with operations to update that record safely (that is, to prevent race situations) when units are acquired or become free, and, if required, wait until a unit of the resource becomes available. This record is paired with operations to adjust that record safely (that is, to avoid race conditions). One further helpful approach to think about a semaphore is as a record of the quantity of a certain resource that is accessible.

Using semaphores is a helpful strategy for the prevention of race situations; nevertheless, the use of semaphores does not necessarily guarantee that a program is free from these concerns. Using semaphores is a technique for the prevention of race situations. Counting semaphores are a special kind of semaphore that can keep track of an arbitrary number of resources. On the other hand, semaphores that can only take on the values 0 and 1 (also known as locked and unlocked, unavailable and available) are known as binary semaphores, and they are the kind that are used to construct locks.

Edsger Dijkstra, a Dutch computer scientist, is generally regarded as the person who first conceptualized the idea of a semaphore in either 1962 or 1963[2]. During this time period, Dijkstra and his coworkers were engaged in the process of developing an operating system for the Electrologica X8 computer. After considerable time had passed, that technique eventually became generally acknowledged as THE multiprogramming system.

3.3 PIPES

The use of several separate channels to imitate a single pipe through which flow may only proceed in one direction. A pipe consists of two distinct channels, referred to respectively as the sink channel and the source channel. It's possible for data to be written to a sink channel, and it's possible for data to be read from a source channel. Following their delivery to the sink channel, bytes can be recovered back from the source channel in precisely the same sequence in which they were delivered. This prevents the data from becoming corrupted in any way.

It is possible for a thread that is sending data to a pipe to wait until another thread receives those bytes or some other bytes that have been written to the pipe in the past from the pipe. This wait can be initiated by the thread that is sending the data. Nevertheless, its behavior is reliant on the system, and as a result, it is unpredictable. It is essential to keep in mind that although while the majority of pipe implementations will have some amount of buffering between the source and sink channels, one should not assume that this functionality is present just because it is there in the pipe implementation.

The following is a list of things that may all be referred to by the term "pipe":

1. In the context of computer memory, a pipe is a part of memory that is not permanent and can connect two or more processors to improve the overall performance of the computer. Pipes can also be used to transfer data between the processors.
2. The wire that is used to send and receive considerable amounts of data for one or more users who are connected to the internet is referred to as a pipe, which is a phrase that refers to the wire in question. A person who owns a substantial quantity of available bandwidth is said to have a "huge pipe" in Internet slang. The word "pipe" appears frequently in the names of companies that offer Internet connection services, such as FatPipe. Each of our backbone and trunk websites provide additional information that is available for your perusal.
3. The "|" key on a computer keyboard is in the shape of a pipe, which is also widely referred to as a vertical bar. This design was chosen because of its versatility. The presentation of it will be interrupted sometimes for pauses of this kind. In a QWERTY keyboard used in the United States, this sign can also be seen within the backslash key.

3.3.1 Where exactly on the keyboard does one find the pipe button?

The following image, which was captured from a computer, features a screenshot that highlights the "pipe" key on the keyboard of a computer by coloring it red. In addition to that, it is not unheard of for this key to be positioned just below the Backspace key and just above the Enter key.

3.3.2 Instructions for adding the | character to a keyboard in the United States

The symbol for the pipe and the character for the backslash are both found on the same key on English-language PC and Mac keyboards. It may be found to the left of the Enter key, which is also referred to as the Return key. It can be found to the right of the Backspace key. When the Shift key is pressed and held down, the character | collaborates with the Shift key to form a pipe.

Instructions on how to add the pipe symbol (|) to a device that is intended for mobile usage. Launch the keyboard application on your tablet or smartphone, choose either the symbols (sym) or the numbers (123), then select either the (#+=) or the (|) symbol, and finally tap the key that corresponds to that symbol.

3.3.3 What is the purpose of the component of a computer known as a pipe, and how does it work?

Programming in the sense that it is utilized nowadays using computers. The logical OR operator is represented by the boolean operator "||" in computer programming. This is because "||" is a binary operator. When one or more of the requirements for the operands or values are met, the OR operator, which is a form of Boolean operator, will carry out the action or duty that has been assigned to it. For instance, even if the sum of one and one does not equal three, it is nevertheless accurate to say that the sum of one and one equals two or that the sum of one and one equals three. When interacting with languages or utilizing services that do not recognize it as a legitimate operator, the word "or" is frequently substituted with the pipe symbol. This is because the pipe symbol has the same meaning as the word "or."

3.3.4 A delimiter in the form of a pipe that is being utilized at this time.

In addition, the use of one or more pipe characters as a delimiter inside a text file is a standard practice that is regularly followed. This is because pipe characters may be

used to separate different parts of the file. The names provided to the characters in text strings known as delimiters are the names given to the characters that physically divide distinct portions of the string. Punctuation marks such as commas, semicolons, quotes, brackets, and pipes, as well as slashes, are examples of delimiters that may be found in written communication. Each separate piece of data is "delimited" by a character that is chosen before the data is recorded by a computer, regardless of whether the data is being saved in a sequential fashion or in the form of a table.

3.3.5 The delimiter or separator for commands

By utilizing the pipe function on the command line, it is possible to redirect the output of one command to the input of another program. The command line is a sort of user interface that may be accessed not through the use of a mouse but rather by the entry of commands into prompts that appear on a screen. This type of user interface is more common in older computer systems. It is also known as the command screen and the text interface in Windows.

Another name for it is the Windows command line. On a computer running Windows, the phrase "C:Windows>" may occur on the command line to indicate the location of the Windows folder, for instance. Depending on the shell, the "%" or ">" sign might be used in its place in Linux and Unix respectively. In a command line operating system, commands are entered using the keyboard rather than the mouse since this type of operating system does not use a graphical user interface. On the other hand, this particular operating system does not make use of a graphical user interface (also known as a GUI).

CHAPTER 4

PROCESS COMMUNICATION

4.1 RACE CONDITIONS

A race condition is an adverse scenario that occurs when a device or system attempts to carry out two or more operations at the same time, but owing to the nature of the device or system, the activities must be carried out in the right sequence in order for them to be carried out successfully. A race condition may be avoided by ensuring that the activities are carried out in the correct order. A device or system is said to be in a race state if it attempts to carry out two or more operations at the same time, which can lead to a number of undesirable outcomes. The competition for a light switch is a simple example of how time may be of the essence. In some home contexts, a single overhead light fixture may have many light switches that are all connected together.

When these sorts of circuits are used, the setting of the switch has no impact on the way the circuit works in any way. When the light is turned on, you may turn it off by moving either switch from the position it is currently in to a different one. In a similar vein, if the light is off and both switches are in the off position, moving any switch from its current position will turn the light on. This is true regardless of which switch is moved. Keeping this in mind, try to visualize what would occur if two different people tried to turn on the light at the same time by using different switches. This might lead to a dangerous situation. It's possible that one order may cancel the other, or that the combination of the two could trip the circuit breaker. Either way, it's feasible that the circuit breaker could be tripped.

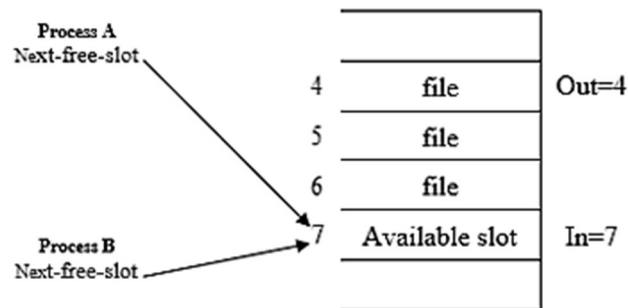


Fig. 4.1

When two or more threads interfere with one another's efforts to finish their respective jobs, this is known as a deadlock. The majority of the time, this problem is brought on by a number of different threads vying for a finite quantity of shared resources... When two distinct threads interact in a negative (buggy) fashion, the nature of that interaction is reliant on the precise order in which each of the threads' different instructions are carried out. This gives birth to a race scenario. It's possible that this will make the software unreliable.

4.2 CRITICAL SECTION

A section of code is considered to be essential if it can be accessed in parallel by many processes. The portion becomes unstable as a result of this type of access. It is possible that the important section will contain shared variables or resources that will need to be synchronized in order to maintain the consistency of the data variables. A critical section is a group of instructions or statements, or a sector of code, that must be done atomically in order to finish the work at hand. For example, accessing a resource (file, input or output port, global data, etc.) is an example of a task that falls under the category of a crucial section. A vital section is a portion of code, to put it another way.

In concurrent programming, the outcome of a "data race across threads," in which one thread seeks to change the value of shared data at the same time that another thread attempts to read the value of the data, may be extremely unexpected. This type of race occurs when one thread attempts to change the value of shared data at the same time that another thread attempts to read the value of the data. It is necessary to synchronize access to shared variables, such as shared memory, shared files, and shared ports, among other things. Only a select few programming languages have native support for synchronization in their core functionality. It is very required to have a strong grasp of the relevance of race situations in order to create kernel mode programming (such as a device driver, kernel thread, or any number of other similar things). assuming that the programmer has unrestricted access to the kernel data structures and the ability to make changes to them.

The following is an explanation of a potential solution to the problems that have been brought up by the essential section: get a hold of the Lock() function; In order for a thread to be able to carry out the instructions of a critical section, it must first acquire a lock so that it can deal with the releaseLock() function that is included within the critical section. Only one of the threads has a chance of obtaining the lock on the door effectively.

4.3 MUTUAL EXCLUSION

When many processes are being carried out at the same time, it is imperative that each process enter the essential part of the program—that is, the section of the program that is shared by all of the processes—at the precise times when it is time to carry out that component of the program. Because many actions are being carried out at the same time, it is likely that the values that are being saved in the essential area may become inconsistent. To put this another way, the values are dependent on the sequence in which the instructions are carried out, and this is what is referred to as a race condition.

The primary purpose of process synchronization is to get rid of race situations that might occur when the most important part of a process is being completed. This is primarily achieved through the use of the principle of reciprocal exclusion. "no two processes can exist in the critical section at any given point in time," which is the definition of the property known as "mutual exclusion," which is a component of process synchronization. "no two processes can exist in the critical section at any given point in time," is the definition of the "mutual exclusion" property.

Dijkstra is credited as being the first person to come up with the term. Any technique of process synchronization that is implemented must be compliant with the property of mutual exclusion in order to function properly. In the event that this is not the case, it will not be possible to eradicate the precarious racial situation that has arisen. (Example) At the same moment, two different customers are browsing for apparel in the clothing section of the same supermarket. Boy A first makes a buying selection, and then he goes to the dressing room to try on the clothing that he wishes to acquire. Because person A is now present in the locker room, the 'occupied' sign that was just affixed to the door signals that no one else may access the space at this time. Because person B also needs to use the changing room, that person is forced to wait until person A has finished their business in the changing room before using the changing room themselves.

As soon as individual A leaves the locker room, the sign on the door changes from reading "occupied" to "vacant," indicating that it is now free for use by a second person. As a direct consequence of this, individual B makes use of the stall despite though it is now labeled as "occupied." The note that was left outside the changing room outlines the process synchronization approach that is now being employed. The changing room is actually only the fundamental component of the whole thing, person A and person B

are both their own independent processes, and the notification was left outside the changing room.

4.4 HARDWARE SOLUTION

The word "hardware solution" refers to the usage of physical devices and components to carry out particular responsibilities or processes that are required by an operating system (OS) or the programs that run on it. When referring to an operating system (OS), the phrase "hardware solution" refers to this use.

4.4.1 The Importance of Hardware Solutions in the Construction of Operating Systems

- Permits the operating system to connect to and interact with physical components, including input/output devices, storage devices, and networking components. Hardware solutions are required for an operating system to connect to and interact with physical components such as input/output devices, storage devices, and networking components. These capabilities are dependent on the capabilities of the operating system.
- Runs faster and more efficiently than it would if some activities were done purely by software because the operating system is able to offload them to the associated hardware components. This makes it possible for the OS to handle more tasks at once. Because of this, the operating system is able to work more rapidly and efficiently than it would be able to if this were the case.
- Helps to ensure that the system is reliable and consistent. By offering specialized components that are built particularly to carry out the functions that the solution is meant to better, hardware solutions can assist in increasing the reliability and consistency of an operating system. This is accomplished by delivering the improvements that the solution is intended to make.
- **Facilitates interoperability with a wide range of hardware combinations, including:** By making use of standardized hardware components and drivers, an operating system can make it easier for the system to be compatible with a wide variety of configurations for the computer's hardware. Users will find it much simpler to upgrade or replace components, and the system will be compatible with a wider variety of configurations as a result.

- **Offers assistance for the increased functionality of the system:** The capacity to implement sophisticated features like graphics processing, virtualization, and encryption can be granted to an operating system by means of the use of hardware solutions. The fact that these functions are supported is what allows for this advantage to be obtained.
- **Contributes to the efficient use of resources by:** An operating system can potentially increase the performance of a computer by making the most of the resources offered by the hardware, such as graphics cards and specialized storage units. Optimization of the available resources is the key to achieving this goal.

4.4.2 Various sorts of hardware solutions that are accessible within the operating system

- A. Drivers for the Various Hardware Devices:** Because of the hardware drivers that are included in software packages, an operating system is able to interface with and manage a wide variety of different types of hardware devices. They are responsible for converting the instructions that are sent by the operating system into directives that the hardware is able to comprehend and carry out. Because of the graphics card driver, an operating system, for example, will be able to show both still images and moving video on a monitor. This capability is not limited to only still images, though. The majority of the time, the drivers for the hardware are already included inside the operating system itself. Alternatively, the drivers may be obtained straight from the manufacturer's website.
- B. The System Software:** A motherboard, a hard drive, or a network card are all examples of hardware components that might have software built into them. This type of software is referred to as "firmware." It is the responsibility of this component to manage hardware-level activities and to supply the device with the instructions it requires in order to work in the correct manner. In a manner that is equivalent to that of software, firmware is not loaded into memory when the system starts up, nor is it saved on a hard drive. This is done so that it can function in a manner that is analogous to that of software. Instead, data is stored on a chip or some other sort of non-volatile storage medium that is included within the actual hardware of the device.

C. There is the C. BIOS: The firmware known as the Basic Input/Output System, or BIOS, is responsible for initializing and configuring the different pieces of hardware during the part of the process known as booting. This phase is also known as "booting up." The operating system is able to communicate with several pieces of hardware thanks to the low-level interface, such as the keyboard, the mouse, and the hard drive. When the computer is switched on, the BIOS is normally read from the Read-Only Memory (ROM) chip located on the motherboard. This allows the BIOS to be loaded into memory as quickly as possible. The control of the system is then handed over to the operating system once a series of tests have been run to validate that all of the components of the hardware are running in accordance with the specifications.

D. UEFI, circa a.D: Most current computer systems make use of a newer firmware standard known as UEFI, which stands for Unified Extensible Firmware Interface. The earlier firmware standard known as BIOS was known as the Basic Input/Output System. It provides a more complicated interface for setting the different hardware components when the system is starting up, and it supports more current features such as secure boot and faster startup times. In addition, startup times are reduced. The UEFI is loaded into memory after being read from a flash memory chip that is situated on the motherboard. This process occurs whenever the machine boots up. In addition to being interoperable with modern hardware and operating systems, it offers an interface that is both more flexible and changeable than the BIOS was originally designed to be.

4.5 THE PRODUCER CONSUMER PROBLEM

The Consumer and the Producer both have their own set of steps that they need to follow. When it comes to the consumption of the things that are produced by the producer, the responsibility is with the consumer. Both the Producer and the Consumer processes make use of the buffer, which may be thought of as a shared space or memory region. The item that was produced by the Producer is stored in the buffer, and the Consumer will process the item from the buffer if it is necessary. The buffer is where the item is maintained after it was made by the Producer. There are two distinct manifestations of this issue: the first is known as the unbounded buffer problem, and it is characterized by the fact that the Producer may continue to produce items despite the fact that there is no upper limit to the size of the buffer; the second is known as the

bounded buffer problem, and it is characterized by the fact that the Producer may produce a limited number of items before beginning to wait for the Consumer to consume them. Both of these manifestations of the issue are referred to as buffer problems.

The problem of the bounded buffer is going to be discussed in more detail. After the Producer and the Consumer have worked together to create a common history, the Producer will go on to the next step of the process, which is the actual manufacturing of the items. If the total amount of the good produced is identical to the size of the buffer, then the producer will wait for the consumer to consume it before starting production. If the amount of the good produced is less than the size of the buffer, then production will continue as normal. In a similar fashion, the consumer will first check to see if the goods is still available for purchase at the given time. If the thing that the customer wants is not currently available, they will wait for the manufacturer to develop it if that is the option available to them. Consumers will consume the items if they can get their hands on them and there are enough of them.

4.6 SEMAPHORES

In order to solve the critical section problem, integer variables referred to as semaphores are utilized, and this is accomplished by the utilization of two atomic operations referred to as wait and signal. The synchronization of the process depends on these actions being completed in the correct order.

Some definitions of the phrases wait and signal may be found in the following passage:

- **Wait:** The wait action will produce a negative change to the value of the input S if that value currently has a positive sign attached to it. The operation will not be carried out if S is either negative or zero, as that is not a valid state for it.

wait (S) while (0 S) is equal to S; S--; Si

- **Show some kind of indication:** The signal action, which looks like this:, raises the value of the parameter S, which results in the following: signal(S) S++; signal(S) There are several distinct types of semaphores. The most common types of semaphores are those that count, also known as counting semaphores, and binary semaphores, also known as binary semaphores. The following is an outline of the particulars of these.

- **Maintaining a watchful eye on the semaphores:** These are semaphores, and their value domain is not restricted in any manner, despite the fact that their value is an integer. The count of the semaphores is a representation of the number of resources that are now accessible. These semaphores are used to coordinate access to the resources, and the count of the semaphores. The number of semaphores is automatically raised if additional resources are brought into play, and vice versa: once additional resources are taken out of play, the number of semaphores is immediately lowered.
- **Semaphore signals:** In binary form Binary semaphores are quite comparable to counting semaphores, with the primary difference being that the values of binary semaphores may only take on the values 0 or 1. When the semaphore is in the position 1, only the wait operation will be successful. On the other hand, the signal operation will be successful when the semaphore is in the position 0. Binary semaphores, as opposed to counting semaphores, have the potential to be implemented in a manner that is both more expedient and less complicated.
- **The Advantages Obtained by Utilizing Semaphores:** The following is a list of some of the advantages that can be gained by using semaphores:
 1. Because of the semaphores, only one process at a time may enter the essential area of the system. In addition to strictly adhering to the principle of mutual exclusion, certain methods of synchronization are also noticeably more successful than many of the others. This is in contrast to the majority of the other methods, which do not adhere to this principle.
 2. There is no resource wasting that happens as a result of busy waiting in semaphores since there is no superfluous loss of processor time to verify if a condition is fulfilled to allow a process to access the essential region. As a result, this prevents the use of resources that are not essential.
 3. The implementation of semaphores takes place in the part of the microkernel's code that is not dependent on the machine. They are not reliant on any one specific piece of equipment.
- **The problems associated with the use of semaphore signals:**

The following is a list of some of the disadvantages that are connected with the use of semaphores:

1. Because semaphores are very complicated, the wait and signal operations have to be implemented in the correct order in order to avoid deadlocks.
2. It is not suggested to employ semaphores on the final scale since doing so will lead to a loss of modularity, and this scale is the one currently in use. This is because the wait and signal methods do not allow for the creation of an orderly layout for the system. This is the reason for why this is the case.
3. When semaphores are utilized, there is a possibility that a priority inversion may take place. This indicates that lower priority processes may have access to the crucial region initially, while higher priority processes may gain access to the area later.

4.7 MESSAGE PASSING,

The provision of the mechanism known as process communication, which is the means by which programs are able to interact with one another, is within the purview and responsibility of the operating system. This communication may include one process alerting another process that a certain event has occurred or the transmission of data from one process to another. Alternatively, it may involve the transfer of information from one process to another. These two instances are both examples of communication happening between different processes. One of the models that may be utilized in order to represent the manner in which processes communicate with one another is known as the message passing model.

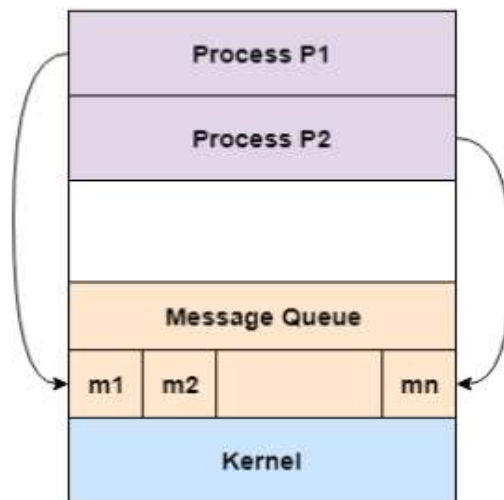


Fig. 4.2

It is possible for separate processes to read and write data to the message queue independently of one another and without the requirement for direct communication thanks to the design that supports message forwarding. The queue is used to store communications in a secure location until the recipient is ready to retrieve them. Because message queues have such a high utility for making it easier for processes to communicate with one another, the vast majority of operating systems make use of them. The figure that follows provides an illustration of a model of process communication that is commonly referred to as the message forwarding model.

According to the graphic that was just presented, the processes P1 and P2 are both able to retrieve and save data from the message queue.

- **The Benefits of Utilizing the Message Passing Model:** The following is a list of some of the benefits that come with using a model that passes messages:
 1. When compared with the shared memory paradigm, the message forwarding approach is significantly simpler to put into action.
 2. Since the message passing paradigm is fairly tolerant of greater communication latencies, it makes it much simpler to construct parallel hardware by utilizing the model.
- **The Message Passing Model Has a Number of Drawbacks:** Communication in the model with message passing is slower than communication in the model with shared memory because setting up the connection requires time.

4.8 CLASSICAL IPC PROBLEMS

4.8.1 Reader's & Writer Problem

The readers-writers problem describes a situation in which one object, such a file, is utilized by many processes at the same time and is consequently shared. Some of these processes are readers, which indicates that all they want to do is read the data that is included within the object, while others of these processes are writers, which indicates that they want to write into the object. The readers-writers problem is employed so that synchronization can be controlled and to guarantee that there are no problems with the object data. For instance, even if two readers access the item at the same time as each other, there won't be a problem because of how the system is designed. On the other hand, there is the potential for problems to arise in the event that the object is concurrently accessed by a reader and a writer, or perhaps by two writers.

To solve this issue, a writer ought to be provided exclusive access to an item; that is, when a writer is utilizing the object, neither a reader nor another writer ought to be able to utilize it at the same time. This is necessary so that the issue may be remedied. On the other side, several users may access the item at the same time, which is really convenient. In order to put this plan into effect, semaphores are a very helpful instrument. Within the context of the reader-writer conundrum, the following is a list of the codes for both the reading process and the writing process. – The responsibility of the reader The reader operation is described by the lines of code that follow: wait (mutex), rc ++, and if (rc == 1) wait (wrt), followed by signal (mutex). READ THE TASK AT HAND. wait (mutex), then send a signal (wrt) if the rc value is equal to zero, and finally send a signal (mutex). The code that was just displayed included semaphores with the names mutex and wrt.

Both of these semaphores had their values set to 1 in the code. In addition, the value of the variable rc is initialized to 0 when the variable is first created. Both the reader and the writer will need to be familiar with the wrt portion of the process code. The mutex semaphore is the component of the process that is accountable for guaranteeing that mutual exclusion is maintained. The value of the variable rc represents the current number of readers who are observing the content of the page. As soon as the value of rc increases by one, the wait operation is carried out on wrt. This implies that a writer will no longer have access to the item after this point in time. After the read process has been finished, rc will be reduced by one value. In the case when rc is equal to zero, the signal action will be carried out on wrt. This indicates that a writer is now permitted to access the item.

The Author's Finished Product or Work This is a listing of the code that specifies the writer process, which may be found below: wrt, then write into the object when some time has passed. wrt; signal; signal(wrt) Whenever a writer makes a request to access an object, the wait action is carried out on the wrt variable. When that time comes, it will no longer be feasible for any other authors to access the object in question. A signal action is performed on the wrt variable whenever a writer has completed writing into an object. This happens whenever the writer has finished writing.

4.8.2 Dining Philosopher Problem etc.

According to the problem of the dining philosophers, there are five philosophers who sit together at a circular table and take turns eating and pondering while they do so.

They do this while conversing with one another. Each of the philosophers is served with a single bowl of rice as well as a set of chopsticks to eat their meal with. When eating, a philosopher is required to place one chopstick on each side of their mouth using both of their chopsticks. A hungry philosopher won't be able to satisfy their hunger if there aren't two pairs of chopsticks available on the table. If this is the situation, a philosopher will set down their chopstick and begin to contemplate once more. The eating philosopher is a well-known example of a synchronization problem that is referred to as a classic due to the fact that it demonstrates a large variety of different concurrency management concerns.

4.8.2.1 Solution of Dining Philosophers Problem

It is possible to find a solution to the "Dining Philosophers Problem" by using a semaphore as an alternative to a chopstick as one of the possible options. It is possible to pick up a chopstick by performing a wait action on the semaphore, and it is also possible to release a chopstick by executing a signal operation on the semaphore. The graphic that can be found below the semaphore chopstick illustrates the components that make up the chopstick. All of the values of the chopstick's components are initialized to one at the start of the process since the chopsticks are still laying on the table at this point and a philosopher has not yet picked them up. The following is a description of the composition of a philosopher that I selected at random: `do wait(chopstick[i]); wait(chopstick[(i+1)% 5]); and EATING THE RICE. signal(chopstick [i]); signal (chopstick [(i+1)% 5]); thinking. while(1)` followed by `signal(chopstick[i])` followed by `while(1)` The first wait operation is carried out on the variables `chopstick[i]` and `chopstick[(i+1)% 5]` in the structure that was explained before. This suggests that the philosopher `i` has set the chopsticks to the side in a separate location. The process of eating then takes place after that point in time.

Following that, signal operations will be performed on both the `chopstick[i]` and `chopstick[(i+1)% 5]` signals respectively. The fact that the philosopher `i` has placed the chopsticks on his sides suggests that he is finished eating, as can be observed from the previous sentence. After then, the philosopher goes back to thinking deeply about the issue at hand.

- **Concerns regarding the remedy that was suggested:** If you use the strategy that was just described, you can be assured that no two philosophers who are sitting next to each other will ever be able to eat at the same moment. However,

there is a risk that this approach may result in a standstill in the discussion. This is something that may take occur if all of the philosophers select their left chopstick at the same time. Because of this, none of them are able to eat, which brings to a stalemate in the situation.

The following is a summary of some of the methods that may be utilized to break free of a stalemate:

1. There should be no less than three and no more than four philosophers sitting at the table.
2. The philosopher with an even number should pick the left chopstick first, followed by the right chopstick, whereas the philosopher with an odd number should pick the right chopstick first, followed by the left chopstick.

Waiting until both sets of chopsticks are available at the same time is the best strategy for a philosopher who needs to make a decision between more than one pair of chopsticks.

CHAPTER 5

DEADLOCK

5.1 PRINCIPLES OF DEADLOCK AND STARVATION

Every procedure calls for a specific quantity of resources in order to be carried out to its greatest potential. However, access to the resource is granted in a certain sequence that has been set in advance.

1. The processing of the requests for a certain resource is now underway.
2. If the resource is accessible, the operating system should make access to it available to the process; if it is not, the process should be permitted to wait.
3. The method takes use of it and then lets it go after it's finished with it.

A situation that is known as a deadlock happens when every computer process waits for a resource that is currently being used by another process. This might lead to a situation where the computer is unable to complete any tasks. In this situation, none of the processes are carried out since the resource that they need is being utilized by another process that is waiting for yet another resource to become available. This means that none of the processes can be carried out. Because of this, it will not be possible to carry out any of the processes.

Let's pretend for a second that there are three distinct processes, each of which is called P1, P2, and P3. The three unique resources that are accessible are denoted by the letters R1, R2, and R3. It has been decided that R1 will take the position at P1, R2 will take the position at P2, and R3 will take the position at P3.

After a certain length of time, P1 places an order for R1, which is then employed by P2. P1 halts its operations since it is impossible to proceed without R2, which is required for completion. In addition, P2 makes inquiries concerning R3, which P3 is now putting to use. P2 likewise comes to a halt with its operations since it is unable to continue functioning in the absence of R3. The operation of P3 has also come to a standstill because of its requirement for R1, which is now being used by another process (P1) in the system.

Within the parameters of this circumstance, the three actions are beginning to form a cycle with one another. Everyone is basically sitting about doing nothing while they wait for anything to take place because nobody is making any progress. The computer will no longer comply with your instructions since all operations have been paused at this point.

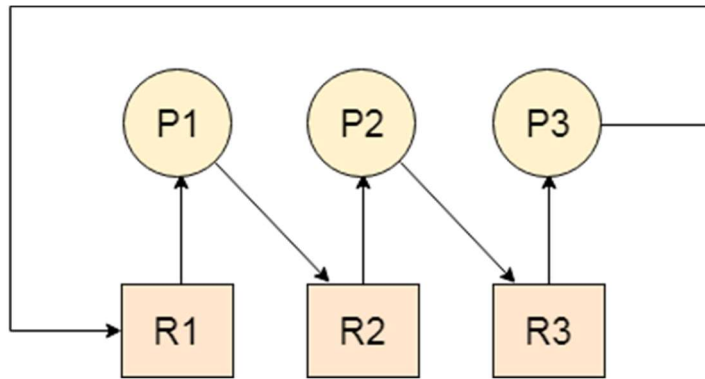


Fig. 5.1

Difference between Starvation and Deadlock:

Sr.	Deadlock	Starvation
1	Deadlock is a situation where no process got blocked and no process proceeds	Starvation is a situation where the low priority process got blocked and the high priority processes proceed.
2	Deadlock is an infinite waiting.	Starvation is a long waiting but not infinite.
3	Every Deadlock is always a starvation.	Every starvation need not be deadlock.
4	The requested resource is blocked by the other process.	The requested resource is continuously be used by the higher priority processes.
5	Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously.	It occurs due to the uncontrolled priority and resource management.

5.2 DEADLOCK PREVENTION

If we wish to replicate a deadlock with a table that is standing on its four legs, then we may also simulate those four legs with the four conditions that, when they occur concurrently, cause the deadlock.

On the other hand, if we are successful in tearing off one of the table's legs, the table will most likely topple over. The same reasoning applies to the stalemate situation; we might be able to escape it if we can violate any one of the four essential conditions and prevent all of them from occurring at the same time.

Let's take a look at the ways in which we can prevent each of these potential outcomes from happening.

1. Exclusion by Consent of Both Parties:

When seen from the point of view of the resource, the concept of mutual exclusion describes the reality that a resource can never have access granted to it by more than one activity at the same time. The fact that this restriction is in place, despite the fact that it could make sense, is the fundamental factor that has led to the impasse. If it were possible for several processes to make use of a resource at the same time, then the process in question would not have had to wait for any of the resources to become accessible. However, if we are successful in penetrating resources that are operating in a manner that is mutually exclusive to one another, then we will be able to break the impasse and go on to the next phase.

- **Getting onto a roll:** The spooling method is one that might come in handy for many devices, such as printers. The tasks that are produced by each of the processes are saved in the memory that is attached to the printer. After a certain amount of time has gone, Printer will collect all of the jobs and print them one at a time using the first-come, first-served technique. Due to the fact that the process takes use of this method, it is not necessary for it to wait for the printer; rather, it may continue doing whatever it was doing before. In a later stage, it gathers the output as it is being produced. This occurs at a later stage.

Spooling is plagued by two different kinds of problems, despite the fact that it has the potential to be an effective approach for getting over mutual exclusion. These problems may be broken down into two categories.

- This does not apply to all of the resources that are currently available.
- There is a possibility that, at some time in the future, there could be a circumstance in which the processes will be competing against one another in order to acquire space in that pool.

We can't make it a requirement that many processes use the same resource at the same time since it wouldn't be logical or fair, and it may lead to significant problems with the processes' ability to carry out their tasks. Because of this, we are unable to make an exception to the rule that a process must adhere to the principle of mutual exclusion.

2. Keep your cool and continue to attend:

The situation that is referred to as "hold and wait" takes place when a process maintains possession of a resource while waiting for the completion of the task being done by another resource. It is conceivable for many processes to be clinging to the same resource at the same time while they are waiting for the next step in the cyclic cycle, which can result in a condition known as a deadlock.

However, in order to verify that a process either does not wait for any resources or does not maintain any resources, we need to discover a mechanism that permits either of these outcomes. This will allow us to ensure that the process in question does not wait for any resources or keeps any resources. This suggests that a process must first be supplied with all of the necessary resources before the execution of the process can get underway. It is imperative that once a process's execution has commenced, there be no waiting for any resources in order for it to continue.

Hold or wait is the same thing, so don't worry about confusing the two. (the disadvantage of using hold and wait is that you have to choose between not holding or not waiting.) If a process is transparent about all of its available resources before the execution begins, it may be possible to achieve this goal successfully. However, despite the fact that this looks to be a very working solution, it is not feasible to implement it in the computer system since a process cannot first identify which resources are necessary. Despite the fact that this appears to be a highly workable solution, it is not possible to apply it in the computer system. The collection of directives that are carried out by the central processing unit (also known as the CPU) is referred to as a process. Each of the instructions may require a different set of resources at a different moment

depending on the specifics of the situation. The requirements of the need cannot be fulfilled by the operating system.

The following is a list of some of the problems with the solution that has been proposed:

1. It is not possible to carry out in actuality.
2. As a result of the fact that certain processes might maintain a resource for an unusually lengthy period of time, the probability of being malnourished will increase.
3. The Lack of Any Kind of Preemption

It is not feasible to halt the progression of a process after it has been started, which may result in a scenario in which there is a standstill. However, if we remove the resource that is preventing the process that is creating the deadlock from continuing, we will be able to get out of the impasse and avoid the deadlock.

This is not at all a good technique given that if we remove a resource that is now being utilized by the process, then all of the work that the process has done up to this point may become inconsistent. In other words, if we remove a resource that is currently being employed by the process, then the process may become inconsistent. Take into consideration the fact that each and every process requires the use of a printer. If we remove the printer from that process and assign it to another process, then all of the data that has been printed has the potential to become inconsistent and ineffective as a result of this change. In addition to this, the process cannot begin printing again from the point where it was stopped, which results in inefficiencies in terms of performance.

3. People Seated in a Circle Waiting:

We have the option to provide a priority number to each of the resources in order to circumvent the cyclic wait. It is against the rules for a process to make a request for a resource that has a lower priority than itself. Because of this, it is not feasible for a single process to submit a request for a resource that is currently being utilized by another process. This eliminates the possibility of a cycle being created as a result of the situation. The only plan that can be put into action is the one that includes violating the circular wait rule, which is one of the several approaches. This is because it is the only strategy that can be done.

5.3 DEADLOCK AVOIDANCE

In order to avoid deadlock, each request for a resource will be granted as long as the resulting state of the system does not cause stalemate in the system. It is planned to perform ongoing monitoring of the system's state, during which they will search for both safe and hazardous circumstances.

It is necessary for the process to tell the operating system of the maximum amount of resources it is able to demand in order for its execution to be finished in order to avoid deadlocks. The most easy and useful technique is that the process announce the maximum quantity of resources of each type that it can potentially demand at any given moment in time. This strategy is recommended since it is the most straightforward and helpful strategy. The approach for avoiding deadlocks explores the allocation of resources in order to ensure that there will never be an instance in which there is a situation in which there is a never-ending cycle of waiting.

It is possible to characterize the resource allocation state of a system by the instances of accessible resources, the instances of assigned resources, and the maximum instance of the resources that are being required by the processes. This may be done by counting the number of accessible resources, assigned resources, and maximum instances of necessary resources.

The following diagram is a representation of the state that a system was in when it was sampled at a moment in time that has not been stated.

Impact on the Available Resources:

Process	Type 1	Type 2	Type 3	Type 4
A	3	0	2	2
B	0	0	1	1
C	1	1	1	0
D	2	1	4	0

Resources still needed:

Process	Type 1	Type 2	Type 3	Type 4
A	1	1	0	0
B	0	1	1	2
C	1	2	1	0
D	2	1	1	2

1. $E = (7\ 6\ 8\ 4)$

2. $P = (6\ 2\ 8\ 3)$

3. $A = (1\ 4\ 0\ 1)$

The resource allocation status of a system may be understood by referring to the tables and vectors shown before, which are denoted by the letters E, P, and A respectively. There are four different processes that make up a system, and there are also four different kinds of resources that make up the system. Table 1 provides an outline of the instances of each resource that are allocated to the various processes.

Table 2 contains a listing of the specific instances of the resources that are still required for each activity. You may look at vector E to find a representation of the total number of instances of each resource that are currently present in the system. The vector P is used to represent the instances of resources that have been assigned for usage by processes. These resources can only be used by those processes. The values in vector A represent the number of available resources that are not being exploited to their full potential at the present time.

It is possible to say that a state of the system is safe if it is able to securely distribute all of the resources that are being sought by all of the processes without coming close to entering a condition known as a stalemate. When the system is in a condition that is considered unsafe because it is unable to fulfill the requirements of all of the processes running on it, the status of the system is said to be hazardous.

When a request is made for resources, it must be granted only under the condition that the state that is reached as a consequence is also a safe state. This is the most crucial part of the strategy for preventing deadlock, and it must be adhered to anytime a request for resources is made.

5.4 DEADLOCK DETECTION

When all of the resources share a single instance, the system is said to have reached a stalemate when a cycle begins to emerge in the graph that represents resource allocation. In the case of a resource allocation graph that contains multiple instances of the same type of resource, cycle is a condition of stalemate that is required, but it is not a condition that is sufficient.

The example that is shown includes three processes that are designated P1, P2, and P3 and also has three resources that are labeled R2, R3, and R3. Every one of the resources has exactly one instance of itself stored in the database. A thorough examination of the graph will reveal to us that it possesses a cycle in its structure. This is because the system meets all four of the traits that are required for a stalemate to take place, which explains why it has occurred.

A Matrix of Assigning Responsibility: One of the jobs that may be performed with a system is to construct an allocation matrix by employing the system's resource allocation graph. There will be a distinct entry made in the Allocation matrix for each of the resources that are distributed in accordance with the distribution plan. For example, in the matrix that is about to be shown, an entry is going to be added in front of P1 and below R3 due to the fact that R3 is identified as being a part of P1.

Process	R1	R2	R3
P1	0	0	1
P2	1	0	0
P3	0	1	0

Requests Arranged in a Matrix: Each of the resources that are being requested will have its own individual entry inserted into the request matrix after it is finished being

constructed. As a result of this, an entry is being formed in front of P1 and below R1, as can be seen in the following example, since P1 requires R1.

5.4.1 Deadlock Detection and Recovery

Within the context of this procedure, the operating system does not make use of any strategy to avoid or prevent the occurrence of deadlocks. As a direct consequence of this, the computer has arrived at the conclusion that the impasse will unquestionably take place. The operating system will frequently check to see if there are any deadlocks in the system and remove them if they are found. This makes it possible to break through impasses. If it finds any of the deadlocks, the operating system will make many attempts to recover the system using a variety of different recovery procedures in the case that it finds one of the deadlocks.

The operating system's principal task is to eliminate deadlocks whenever they arise. The operating system is able to determine whether a stalemate has occurred with the assistance of the Resource allocation graph. If the system is in the process of establishing a cycle at the present, then it is inevitable that there will be a standstill with regard to resource categories that each only have one occurrence. On the other hand, locating a cycle is not adequate in a graph that includes a great number of occurrences of the same kind of resource. The safety algorithm needs to be implemented to the system, and the first step in doing so is to convert the resource allocation graph into an allocation matrix and a request matrix. After that, the algorithm needs to be applied to the system.

Either the system's resources or its operations will be investigated by the operating system in attempt to break the impasse that has been created to a safe situation that existed before. The system goes through a number of distinct stages until it reaches the point where it is in a state of stalemate. The operating system provides the capability to restore the system to a safe and secure condition from a point in time that was previously preserved. In order for the operating system to be successful in reaching this objective, check pointing needs to be implemented at each state.

As soon as we reach a point when there is no clear path forward, we are going to reverse all of the allocations so that we can go back to the safe place where we were before.

- **Put a stop to an ongoing procedure:** It's possible that killing one of the processes may fix the issue, but the more immediate challenge is figuring out

which of the processes should be killed. In the vast majority of instances, the operating system will put an end to a process when it has finished the minimum amount of work necessary up to this point.

- **Stop following each and every process:** This tactic cannot be recommended, but it is an option that may be utilized in the event that the difficulty level of the issue significantly increases. If you terminate every process in the system, the system's overall efficiency will decrease since every process will have to start operating from the very beginning once more.

CHAPTER 6

MEMORY MANAGEMENT

6.1 MEMORY MANAGEMENT REQUIREMENTS

Memory management keeps a record of the state of every memory location, including a notation of whether or not the space has been allocated at the moment. It dynamically allocates memory to the programs that ask for it, and when that memory is no longer needed, it makes it available for other programs to use. This happens at the request of the applications. Memory management that is tailored to meet a variety of requirements, all of which must be kept in mind here.

The following are some of the necessary components for efficient management of memory:

Because the available memory in a multiprogramming system is often shared among a number of processes, it is not possible to know in advance which other programs will be running in main memory at the time that this program is being executed. This is because it is not practical to predict in advance which other programs will be resident in main memory at that moment. The method that is used to transfer this memory is referred to as relocation. The process of moving active processes in and out of main memory makes it feasible for the operating system to have a larger pool of processes that are ready to be executed. This is made possible by the practice of swapping active processes in and out of main memory.

It is not always possible for a program to occupy the same memory address when it is swapped back into main memory after it has been moved out of main memory and into disk memory. This occurs when a program is moved from main memory to disk memory. When the application is swapped back into main memory, the memory location can still be being used by another process. This is the reason for this behavior. It is possible that we will be need to relocate the process to a different part of memory at some point. As a consequence of this, there is a possibility that the program will experience main memory relocation as a direct consequence of swapping.

1. The image displayed in the illustration is meant to depict a procedure. At this point in time, the process image is continuously using a portion of the main memory that

is available. The operating system will require knowledge of a significant variety of information, some of which includes the location of the process control information, the execution stack, and the code input, among other things. Logical addresses are the names given to the memory references that may be found in a program's many different instructions. A program is made up of many different instructions.

Following the loading of the program into the main memory, the central processing unit (CPU) and the operating system need to be able to translate logical addresses into physical addresses. This must be accomplished with flying colors. The branch instructions themselves contain the location of the next instruction that will be executed, which may be retrieved at any time. The instructions for the data reference are formatted so that they include the address of the byte or word of the data to which they are referring.

- 2. Safety and security** – When we run a large number of programs simultaneously, there is always the risk that one of those programs will accidentally write in the address space that is being utilized by another of those programs. This is a danger of which we must always be aware and prepare ourselves accordingly. Because of this, it is imperative that each and every process be protected against unauthorized interference anytime another process attempts to write data into a process, regardless of whether the effort was made accidentally or incidentally. Because it becomes more difficult to satisfy the relocation need once the protection requirement has been completed, there is a trade-off that takes place between the two requirements. The protection requirement must be met before the relocation requirement can be satisfied.

Because it is not possible to determine where a program will be placed in the main memory of the computer, it is not possible to give security by checking the absolute address during the compilation process. This is because it is not possible to foresee where a program will be stored. The great majority of programming languages offer users the ability to dynamically calculate an address while the program is running. Memory protection is a need that must be met by the processor, not the operating system, due to the restricted capacity of the operating system to exercise control over a process when that process is using the processor. Consequently, the memory protection must be given by the processor. As a result of this, it is not difficult to determine whether or not the allusions to memory are authentic.

3. A protection system must contain the feature of sharing in order to let several processes access to the same area of main memory. This is because accessing the same section of main memory might be very useful. It is more efficient to provide each process access to the same copy of the program rather than to give each process its own independent copy. This is because access to the same copy of the program is shared by all of the processes.

For example, many processes may access the same system file at the same time. In this case, it is standard practice to load just one copy of the file into the main memory and to permit the processes to share the resource among themselves. Memory management's primary responsibility is to guarantee that restricted access may be allowed to shared memory areas without in any way compromising the integrity of the data being stored there. It is necessary to make use of various mechanisms in order to permit the capabilities of relocation assisted sharing.

4. **Organization based on logic:** The primary memory can be organized in a linear fashion, or it can take the shape of a one-dimensional address space that is composed of a string of bytes or words. Either way, there are two possible configurations for the primary memory. The great majority of the programs are possible to be broken down into separate modules, some of which are immutable (read-only and execute-only), while others include data that is susceptible to change. Both the operating system and the computer hardware are required to provide support for a core module for it to be possible to effectively interact with user programs. This module has to be able to provide the required level of security while also allowing for adequate sharing. The following is a list of some of the advantages that it provides:

- Modules are written and produced independently, and at run time, it is the responsibility of the "system" to resolve all references from one module to another module. Modules can only refer to other modules in other modules. These references are sent from one module to another via a messaging system.
- Different modules receive varied degrees of protection, according to the requirements that are unique to each module.
- There are ways that enable the sharing of modules amongst a variety of operations. Sharing can be enabled on a per-module basis, providing the user

with the flexibility to choose the level of sharing that is most suitable for the specific module.

5. The layout and organization of the actual area - The framework of a computer's memory is composed of many layers, the first of which is known as main memory. The second layer, known as secondary memory, is located underneath the main memory. Main memory functions at a pace that is substantially faster than that of secondary memory, but it comes at a cost that is significantly higher. The primary memory is not a permanent storage medium. Because of this, secondary memory is made accessible for the storage of data for the long term, whilst main memory is used for the storage of programs that are now being employed by the system. When contrasting main memory with secondary memory, the flow of information is the most important aspect of the system; nevertheless, it is impractical for programmers to understand this for two reasons:

- The approach known as overlaying is one that a programmer may turn to in circumstances in which there is not enough main memory for a program and the data that is connected with it. It enables the simultaneous assignment of a large number of modules to the same area of memory, which was not feasible before. One of its flaws is that it demands a significant investment of time on the part of the programmer.
- The programmer does not know how much accessible memory space there will be at the time of coding in an environment that supports many programming languages, nor does the programmer know where in the memory that space will be located.

6.2 MEMORY PARTITIONING: FIXED AND VARIABLE PARTITIONING

1. **Fixed Partitioning:** Multi-programming with fixed partitioning is a contiguous memory management approach that splits the main memory into fixed sized segments that can be of equal or unequal size. This method is used in conjunction with multi-programming with dynamic partitioning. This method is included in the multi-programming with fixed partitioning approach, which is discussed further down in this section. When we need to allocate memory for a process, we search for a free partition that is large enough to contain the process that we are allocating memory for. When we find a partition that meets these requirements, we then

allocate memory for the process. After this, the RAM will be put to the side for the subsequent operation. In the case that there is not sufficient free space available, the process will wait in the queue in order to be assigned memory. This occurs when there is not enough free space available. It is one of the methods for memory management that has been around the longest and is easy to put into practice. It is also one of the most effective methods.

2. **Multi-programming with Variable Partitioning:** Multi-programming with variable partitioning is a technique for managing contiguous memory in which the main memory is not split into partitions and the process is assigned a piece of free memory that is large enough for it to fit. This approach is used in conjunction with variable partitioning. In this strategy, managing contiguous memory is referred to as variable partitioning. Combining this method with variable partitioning is how it's often implemented.

The term "free space" refers to the area that is left accessible after everything has been taken from it, and it may be utilized in later procedures if necessary. In addition to that, it provides the possibility of making things more compact. During the process of compaction, the memory spaces that are free and the memory spaces that have not been allocated to the process are consolidated in order to generate a single enormous memory space. This single memory space is the result of the compaction process.

Difference between Fixed Partitioning and Variable Partitioning :

S.NO.	Fixed partitioning	Variable partitioning
1.	In multi-programming with fixed partitioning the main memory is divided into fixed sized partitions.	In multi-programming with variable partitioning the main memory is not divided into fixed sized partitions.
2.	Only one process can be placed in a partition.	In variable partitioning, the process is allocated a chunk of free memory.

S.NO.	Fixed partitioning	Variable partitioning
3.	It does not utilize the main memory effectively.	It utilizes the main memory effectively.
4.	There is presence of internal fragmentation and external fragmentation.	There is external fragmentation.
5.	Degree of multi-programming is less.	Degree of multi-programming is higher.
6.	It is more easier to implement.	It is less easier to implement.
7.	There is limitation on size of process.	There is no limitation on size of process.

6.3 MEMORY ALLOCATION: ALLOCATION STRATEGIES (FIRST FIT, BEST FIT, AND WORST FIT)

The techniques that are utilized by operating systems to manage and distribute memory resources to the numerous processes and applications that operate on the system are referred to as "allocation methods." The phrase "allocation methods" was coined in the 1980s. Memory is necessary for the execution of a program since the program needs to store instructions, data, and other resources. Memory can be provided via a computer's hard drive. Memory is parceled out to each process by the operating system, which enables more efficient utilization of the system's resources and helps to ensure their continued availability. The operating system provides users with the option to make use of either contiguous or non-contiguous allocation strategies as the primary

ways of allocation. When a contiguous allocation technique is used in an operating system, memory is distributed to a process in a block that is uninterrupted throughout its whole.

The fixed partition allocation technique is used to partition memory in a fixed manner. This approach then allocates each process to one of the available partitions. When using the dynamic partition allocation method, memory is first divided into divisions of variable sizes. The computer then assigns a partition to each particular process that is perfectly proportionate to the requirements of that process. This method is called dynamic memory partitioning. When the memory for a process is allocated, it is not done so in contiguous blocks as it is when the memory for the operating system is allocated. The technique that locates the largest block of memory that can support the process is referred to as the worst-fit allocation method, whereas the method that locates the smallest block of memory that can support the process is referred to as the best-fit allocation method.

The next-fit allocation technique hands over to the first-fit allocation method the first available block of memory that is large enough to hold the process. Rather of assigning the next available block of memory that can accommodate the process by using the next-fit allocation technique, the first-fit allocation method uses the block of memory that is now available. The specific requirements of both the operating system itself and the programs that are being run on it are taken into consideration when making the decision of which allocation technique should be used in an operating system. Each of these techniques has its own unique set of advantages and disadvantages. It is vital to effectively manage the system's memory resources in order to guarantee the greatest potential performance of the system and the most effective application of its resources. This can only be accomplished through good memory resource management.

- **The many methods of contiguous allocation that are available in the operating system:**

Operating systems frequently implement the contiguous allocation method as one of their approaches to memory allocation. Using this approach, a continuous portion of memory is provided to every process that is operating within the system. Using this method, the memory is first divided into partitions or blocks of a predetermined size, and then a single partition of contiguous memory is assigned to each individual process. This strategy is known as memory partitioning. The concept of contiguous

allocation is not only simple to implement, but it also has the potential to be beneficial when it comes to dealing with relatively insignificant operations. On the other hand, it may cause fragmentation concerns if there is insufficient memory that can be assigned to a process in a single chunk if there is not enough memory available. The overarching concept of "fragmentation" may be broken down into two subheadings, namely, "external fragmentation" and "internal fragmentation."

A process known as external fragmentation takes place when free blocks of memory are spread throughout the memory space. This makes it difficult to identify a block of memory that is continuous for the purpose of assigning it to a process since there are so many different types of memory. When a process is given a block of memory that is larger than what is required of it, a situation known as internal fragmentation might arise since this leaves memory within the block that is not being used. Because of this, the memory allocation that the process uses will be fragmented on the inside.

Compaction and paging are two approaches that are applied by particular operating systems in order to alleviate the fragmentation problem. This problem is caused when data is scattered over several locations on the hard drive. When it comes to the allocation techniques of the Operating System, compaction entails gathering all processes together in order to form a contiguous block of free memory, whereas paging entails dividing memory into smaller fixed-sized pages and allocating non-contiguous pages to processes. Compaction is distinguished from paging by the fact that it produces a contiguous block of free memory. These two approaches are examples of allocation strategies, which is a category that encompasses all of these approaches.

In a nutshell, contiguous allocation is a simple and efficient approach for allocating memory in operating systems, notwithstanding the possibility that it could result in fragmentation issues. Operating systems adopt tactics such as compaction and paging to guarantee effective utilisation of memory resources, which in turn serves to limit the danger of fragmentation. Compaction is the process of packing together smaller pieces of data into larger ones.

- **The Best Possible Fit:**

Memory allocation strategies come in a variety of forms, and operating systems make use of one known as the best-fit allocation approach. In this method, the memory that is allotted is selected from the free partitions that are the smallest that are available

while yet having enough space for the task. When a process makes a request for memory, the operating system searches for the partition that has the smallest amount of free space but is still large enough to accommodate the amount of memory that has been requested. Other techniques of memory allocation have the potential to result in more external fragmentation than the best-fit allocation strategy does. This is due to the fact that the best-fit allocation approach employs small memory free blocks to support smaller activities. This is a perk of the situation. On the other hand, it may also result in a drop in performance since the operating system will need to hunt for the partition that is the most suited for each memory request made by a process. This may cause the performance of the system to fall.

Bin-packing is a strategy that may be used to boost efficiency and is potentially utilized by the operating system. The free memory blocks are partitioned into "bins" of sizes that are consistent with one another and are compared to one another using this manner. Because of this, the operating system is able to quickly identify a free partition that has the capacity to house a process whenever it needs to. The best-fit allocation technique is a memory allocation approach that is used in operating systems. It involves allocating memory in such a way that the smallest free partition that is capable of housing the process is picked for allocation. In a nutshell, the best-fit allocation method is a memory allocation strategy that is used in operating systems. It is possible that as a result of this, the performance of this technique would decline, but it is also possible that it may result in less external fragmentation. The operating system may use an approach known as "bin-packing" in order to boost speed. This strategy involves packing data into bins.

- **The one that fits you the worst:**

The worst-fit allocation technique is a memory allocation approach that is used in operating systems. It is defined by the practice of picking the free partition that has the largest space as the one to be allotted as the memory space for the system to use. When a process makes a request for memory, the operating system will search for the partition that has the largest free space and is capable of housing the process. If it can't be found, the operating system will stop the process from making the request.

The worst-fit allocation strategy has the potential benefit of resulting in less internal fragmentation than other memory allocation algorithms. This is because big free memory blocks are used to support little processes. On the other side, it may also result

in a drop in speed since the operating system will have to hunt for the partition that is the least suited every time a process makes a memory request. This will need the operating system to perform more work than necessary. Because it causes tiny free memory blocks to be left unused, the worst-fit allocation strategy may also result in an increase in the amount of external fragmentation that happens. This is due to the fact that it causes unused free memory blocks to be left. Because of this, the operating system may have a far more difficult time locating free memory blocks of a size that is sufficient to enable the execution of more comprehensive applications.

The worst-fit allocation methodology is typically not used in today's operating systems since it has the potential to result in excessive memory usage as well as lower performance. Consequently, this method is not used. This is due to the fact that it is not utilized very frequently. Other methods of resource allocation in operating systems, such as best-fit, first-fit, or next-fit, are frequently recommended. Examples of these methods include: best-fit, first-fit, and next-fit.

- **The First Attempt at the Experiment:**

Memory allocation methods that are applied in operating systems include the first-fit allocation technique and the first-fit allocation strategy. In this method, the operation of allocating memory is completed by picking the first free partition that is able to take on the process. This strategy is described in more detail here. When a process makes a request for memory, the operating system searches for the first free partition that meets the requirements of the memory requirements made by the process. When adopting the first-fit allocation approach, the operating system does not have to search for the partition that is the best fit or the partition that is the worst fit, which can result in better operational efficiency. The first-fit allocation method has this advantage as one of its benefits. On the other hand, it may also result in a greater degree of internal fragmentation due to the fact that smaller processes may be allocated to bigger partitions, which then results in the partition holding free memory space. This can happen because larger partitions may be given to smaller processes.

When confronted with the challenge of fragmentation, the operating system could choose to employ a tactic known as "splitting" as a method of resolution. In this strategy, a massive partition is divided into a number of more manageable subpartitions so that operations may be performed with more ease. The operating system also has the possibility of employing a process known as coalesce, which includes the merging

of contiguous free partitions into a single larger partition. This process may be carried out by combining free space on many partitions into a single larger partition. The first-fit allocation technique is a memory allocation strategy that is employed in operating systems. It comprises picking for allocation the first free partition that is able to hold the process that is being allocated memory to. In a word, the first-fit allocation method is a memory allocation strategy that is used in computer programs. There is a possibility that using this method may result in enhanced performance; but, there is also a possibility that it will produce an increase in the amount of internal fragmentation. As part of an effort to reduce the level of fragmentation

6.4 DEMAND PAGING

Memory in a multiprogramming system is often segmented into a number of partitions or regions, each of which is allotted to a specific operating process. These partitions and regions may have a fixed or variable size. For instance, a process that requires m words of memory can be executed in a partition that only has n words, provided that n is smaller than m . There is a possibility that the variable size partition strategy will provide an outcome in which the accessible memory is not continuous but rather broken up into a number of dispersed chunks. We differentiate between fragmentation that occurs within and fragmentation that occurs externally. Memory that is internal to a partition but is not being used is referred to as internal fragmentation, and the difference between n and m represents the size of this memory. It is considered to be an example of external fragmentation if a partition is both unused and available but is of a size that prevents it from being used by any waiting process. These pieces of memory are useless and cannot be utilised.

We can either create a paging strategy that enables a program's memory to be non-contiguous, which enables a program to be allocated to physical memory, or we can compress the memory, which will make big memory blocks available for use. Either way, we will be able to tackle this issue.

The physical memory is segmented into frames, which are blocks of a consistent size. Pages are units of logical memory that are uniformly sized blocks and are organized in a hierarchical structure. When it comes time to run a program, the pages of that program are read from the disk and loaded into any memory frames that are free. Additionally, the disk is broken up into blocks of a consistent size, which have the same dimensions as the memory frames.

Paging makes a distinct divide between the memory that the user sees and the memory that is actually there on the device, which is a highly significant feature. In most cases, a user will have the assumption that memory is one continuous region that solely contains their own software. In point of fact, the logical memory is dispersed throughout the physical memory, which also stores a number of different programs. As a result of the address translation or address mapping, the user is able to operate accurately with the view of memory that is specific to them. Address mapping is the process by which an operating system converts logical memory addresses into their corresponding physical addresses. This process is completely hidden from users and is managed by the operating system.

Because the operating system is in charge of managing the memory, it is necessary for it to have complete knowledge of the characteristics of the physical memory. For instance, it must know which frames are accessible and which are assigned, as well as the overall number of frames. All of these parameters are stored in a data structure known as a frame table, which contains one entry for each physical frame of memory and indicates whether it is free or allocated, as well as to which page of which process it is assigned if it is allocated. If it is free, it indicates that it is not allocated.

Because there is a significantly less amount of physical memory than virtual memory, the operating system needs to exercise extreme caution to ensure that it does not use the physical memory in a manner that is wasteful. Saving physical memory can be accomplished in part by loading just those virtual pages that are really being utilized by the application that is being carried out at the moment. For instance, in order to query a database, a database program might need to be executed. In this particular instance, just those data records that are going to be investigated need to be imported into memory rather than the complete database. In addition, if the database query is a search query, it is not necessary to load the code from the database that deals with adding new entries because such code already exists in the database. Demand paging is the term given to the process of only loading virtual pages into memory when those pages are really being used.

The CPU is unable to locate an entry in the page table for the virtual page that is being addressed when a process attempts to access a virtual address that is not currently resident in memory. For instance, in Figure 1, Process X's page table does not have an entry for virtual PFN 2, and as a result, the CPU will be unable to convert a virtual address into a physical one if Process X attempts to read from an address located within

virtual PFN 2. At this point, the central processing unit (CPU) is unable to keep up, thus it is up to the operating system to remedy the problem. It alerts the operating system that there has been a page fault, and the operating system forces the process to wait while it repairs the issue. The central processing unit is responsible for loading the relevant page from the disk image into memory. Because accessing the disk takes a significant amount of time, the procedure must wait for a considerable amount of time until the page has been retrieved.

If there are more processes that might run, the operating system will choose one of them to execute if there are several processes that may run. After the page has been retrieved, it is written into a free physical page frame, and an entry for the virtual PFN is added to the page table for the process. After then, the procedure is started over from the point where the memory error occurred. Now that the access to the virtual memory has been made, the CPU is able to do the address translation, and the process is able to carry on as normal. This phenomenon, which is known as demand paging, takes place not just when the system is extremely active but also while an image is being put into memory for the very first time. Because of this technique, a process will be able to carry out the execution of an image even if it only partially lives in the physical memory at any one moment.

The valid/invalid bit of the page table entry for a page that is being swapped in is set to be valid. This occurs when the page is swapped in. If that doesn't happen, it gets marked as invalid, which won't matter as long as the software doesn't try to visit this page in the first place. The procedure will carry out in the same way as if all of the pages were brought in if all of the necessary pages, and just those pages, are switched in. If the process attempts to access a page that has not been swapped in, and the valid/invalid bit of this page table entry is set to invalid, then a page fault trap will occur. This occurs when the process tries to access a page that has not been swapped in. It shows that the operating system was unable to get a legitimate component of the application into memory at the proper moment in order to reduce the expense of swapping in place, rather than displaying the "invalid address error" as it would normally do.

In order for the operating system to continue the execution of the process, it will plan a disk read operation to put the needed page into a freshly allocated frame. This will allow the process to continue. Following that, the record in the associated page table will be updated to reflect the fact that the page is now present in memory. Because the

state of the stopped process (including the program counter and registers, among other things) was stored when the page fault trap occurred, the process that was interrupted can be resumed at the same position and state as before. It has been demonstrated that it is feasible to execute programs even while certain components of the programs are not (yet) stored in memory.

In the worst possible scenario, it is possible to run a process even if there are no pages in memory. The first instruction would result in a page fault trap being generated. Following the loading of this page into memory, the process would proceed to carry out its instructions. Page fault trap would continue to occur in this manner until each and every page that was required was present in memory. The term "pure demand paging" refers to this particular method of pager use. The principle of "never bring a page into memory until it is required" lies at the heart of pure demand paging.

6.5 VIRTUAL MEMORY

Memory Management was one of the topics that were discussed in the first several lessons of this class. Overlays, contiguous memory allocation, static and dynamic partitioned memory allocation, paging schemes, and segmentation techniques were some of the subjects that were discussed during the course. In this session, one of the most important topics that we will discuss is memory management, and more specifically, a subject known as virtual memory. Storage allocation has always been a major challenge in computer programming because of the very low cost of secondary storage and the comparatively large quantity of secondary storage. The reason behind this is because the cost of main memory is quite high, but the cost of secondary storage is just moderately expensive. The main memory is the location where all of the data and the computer code that is required for the successful execution of a process must be stored.

However, there is a possibility that the main memory does not have sufficient space to handle all of the needs that are essential for a complete procedure. The first individuals to design programs for computers split them up into chunks that could be temporarily kept in the main memory of the computer while the program was being processed. These fragments were then reassembled back into the original program. As the program ran, new sections migrated into the main memory and replaced sections that were no longer necessary at that time. These replacements occurred as the program progressed. As the program moved on, these substitutions were made as necessary. During this

early period in the history of computers, the man who was responsible for the construction of this overlay system was the programmer.

Because increasingly difficult programs were written in higher-level languages, which became more common as machines became more powerful, and because programmers were less familiar with the machines themselves, the efficiency of complex programs deteriorated owing to insufficient overlay systems. This was mostly caused by the fact that complex programs were written in higher-level languages. The problem of allocating the available space for storage became more difficult to resolve as time went on. Both static and dynamic allocation are schools of thought that have been put up as possible answers to the problem of inefficient memory management.

Both of these techniques have their advantages and disadvantages. The concept of static allocation is predicated on the idea that it is possible to ascertain in advance not only the amount of memory resources that are accessible to a program but also its memory reference string. When dynamic allocation is employed, it is not feasible to predict the amount of memory that will be required by a program since the amount of memory used has to fluctuate according to the requirements of the application. In the 1960s, the program objectives and technical advances that were taking place made it difficult, if not impossible, to create the requisite forecasts for static allocation. This was due to the fact that static allocation required a fixed amount of resources.

As a consequence of this, the solution of dynamic allocation received a great deal of support, despite the fact that opinions about its implementation remained divided. According to one school of thinking, the programmer should be the one to maintain responsibility for the allocation of storage space. This obligation would be carried out by using system calls to either allocate or deallocate memory. The second group argued in favor of the operating system being responsible for automating the process of allocating storage space. This was done as a reaction to the rising relevance of multiprogramming as well as the growing complexity of storage allocation.

In the year 1961, two different companies each showed their version of a memory store that had only one level. One of the suggestions called for an extraordinarily large amount of primary memory to be used in order to do away with the necessity of any storage allocation. Because of the staggering amount of money that would have been required, this strategy was not a viable choice. The virtual memory proposal describes the second concept being discussed here.

- **Virtual Memory:**

It is not unheard of for a modern processor to have the ability to run many distinct processes at the same time. The address space that is associated with a process is one of a kind and is only used by that particular process. When you consider the fact that processes can be started and stopped often, in addition to the fact that many processes utilize only a small piece of the address space that is made accessible to them, it would be prohibitively expensive to build a full address space from scratch for each and every process. This is because many processes use only a small amount of the address space that is made available to them. In conclusion, but certainly not least, it is essential to emphasize that although there have been significant advancements in hardware technology, the resources that may be utilized by machines are still restricted.

Therefore, it is necessary to share a smaller quantity of physical memory among several processes, while simultaneously creating the illusion for each process that it has its own distinct address space. This is because the amount of physical memory available is limited. This is done in order to lessen the amount of space used up by the physical memory. The most popular way to achieve this goal is through utilizing a technique known as virtual memory, which has been around since the 1960s but has only lately become ubiquitous on computer systems. In the late 1980s, virtual memory first started to gain significant adoption. The virtual memory scheme partitions the physical memory into blocks, and these blocks are then parceled out to the many applications that are currently active in the system.

It should go without saying that in order to do this in a logical manner, it is very desirable to have a protection mechanism that limits the capacity of a process to access only those blocks that are expressly given to it. This should go without saying because it should go without saying that in order to achieve this in a decent manner, it is quite desirable to have a protection system. In light of this, a security strategy of this sort is an essential component of every virtual memory implementation, regardless of the level of complexity that must be met. One additional benefit of utilizing virtual memory that may not be immediately apparent is that it typically shortens the amount of time necessary to begin a program.

This is one benefit that may not be immediately visible. This is due to the fact that not all of the program code or data need to be present in the computer's physical memory in order for the program to be able to start being executed. This is a benefit that might

not be immediately obvious to the reader. Although it is a desirable goal, sharing the physical address space is not the main reason why virtual memory has become so commonplace on current systems. Up until the late 1980s, it was the job of the programmer to make sure that a program would not grow to be so huge that it would no longer be able to be stored in its entirety in the physical memory of the computer. This practice persisted until the late 1980s. It was common practice for programmers to achieve this by breaking up programs into pieces, each of which included logic that was incompatible with the logic of the other portions.

When a program was launched, the major component that kicked off the execution would first be placed into physical memory. This would happen every time the program was started. After that, the additional components, also known as overlays, would be loaded into physical memory as and when they were necessary. This process would repeat until all of the components had been loaded. The programmer was responsible for ensuring that the program never attempted to access more physical memory on the computer than was actually available, and it was also the programmer's responsibility to ensure that the appropriate overlay was loaded into physical memory whenever it was necessary to do so.

As a direct result of these duties, numerous obstacles were presented to programmers. They were expected to be able to split their programs into parts that were conceptually separate from one another and to describe an acceptable scheme to load the relevant fragment at the right moment. The ability to break their programs into pieces that were conceptually distinct from one another was a prerequisite. The requirement that programmers working on large software projects be liberated from the laborious and time-consuming chore of generating overlays was a driving force behind the development of virtual memory.

The memory hierarchy consists of two levels, and the automated administration of both of these levels is handled by virtual memory. These tiers consist of Virtual Memory, which stands in for the primary memory, and secondary storage respectively. This management is carried out in a manner that is totally imperceptible to the application that is now being carried out. It is not necessary for the actual execution of the program to worry about the specific physical location of any piece of the virtual address space at any point. The same program may be run from any address inside the physical memory thanks to a process called relocation, which moves the program. This is attainable due to the fact that the software is able to relocate its data in several locations.

Before the development of virtual memory, it was common practice for machines to have a relocation register that was designed for the sole purpose of transferring data from one location to another. This was done before the introduction of virtual memory.

An alternative to the physical solution of a virtual memory that consisted of software that updated all addresses in a program each time it was performed would be an expensive and laborious process. It is also a solution that would create a mess. A solution such as this one would, among other things, significantly increase the length of time necessary for programs to finish their cycles. Because virtual memory exists, a process just has to make an effort to reach any block of its address space; there is no need to care about the location of the block it is trying to access. Because of virtual memory, a software can ignore the location of any chosen block in its address space regardless of where the block really is located on the computer's hard drive. If the block happens to be placed in the main memory, accessing it is simple and quick; otherwise, the program will need to wait for the virtual memory to bring the block in from secondary storage before it is made accessible for access.

Utilizing processor caches in a way that is similar to how virtual memory is used is, in some ways, a practice that is analogous to how virtual memory is used. When compared to the block size of the typical processor cache, which may be anywhere from 128 bytes to 64 kilobytes and up, the block size of virtual memory can be anywhere from 64 kilobytes and up. Virtual memory can also have a block size of 128 bytes and up. This is one of the most important differences that can be drawn between the two kinds of memory. The hit time, the miss penalty (the amount of time added to the total time needed to obtain an item that is not already in the cache or main storage), and the transfer time are all extended when utilizing virtual memory.

The miss penalty is the amount of time added to the total time needed to retrieve an item that is not already in the cache or primary storage. On the other hand, the percentage of shots that are missed is often a great deal lower. (This is no accident—since a secondary storage device, typically a magnetic storage device with much lower access speeds, has to be read in case of a miss, designers of virtual memory make every effort to reduce the miss rate to a level even much lower than that allowed in processor caches). There are two basic kinds of virtual memory systems: those that use pages, which are blocks of a predefined size, and those that use segments, which are blocks of different sizes. Pages are used by systems that use fixed memory sizes, while segments are used by systems that use variable memory sizes. Imagine, for instance,

that there is a requirement for 64 megabytes of main memory but that there are only 32 megabytes that are really available. What would you do? After dividing the required amount of space into portions that are referred to as pages, the memory manager would then store the contents of these pages in the mass storage device. This would give the impression that there was more memory space available than there actually was.

A page can be a maximum of four kilobytes in size before it is regarded abnormally large. The memory manager would switch the pages that are genuinely essential in the main memory with pages that are no longer required, and as a consequence, the other software units would be able to carry out their responsibilities as if the computer had an actual main memory capacity of 64 megabytes. In a word, virtual memory is a technique that permits the execution of programs that might not be totally included in memory. This is made possible by the fact that virtual memory can simulate the contents of physical memory. The possibility that the size of the program will surpass that of the available amount of physical memory is a big advantage of adopting this method. The construction of virtual memory may be done in a couple of different ways, the most common of which being demand paging and demand segmentation.

6.5.1 VM management

This section provides a thorough analysis of how the virtual memory manager enables the supply of virtual memory. This article aims to provide a comprehensive understanding of the correlation between logical and physical address spaces, as well as the situations that necessitate the utilization of services provided by the Virtual Memory Manager. Before going into the subtleties of virtual memory management, it is necessary to first explore the functions executed by the virtual memory manager. The pertinent actions encompass:

- Assigning portions of the logical address space for storage in physical random access memory (RAM)
- Ensuring the immobility of certain segments within the logical address space in physical RAM.

This process involves the establishment of a correspondence between logical addresses and physical addresses. Additionally, it entails the postponement of the execution of interrupt code, which is supplied by the application, until an appropriate moment.

Before considering the many strategies that are used by operating systems to make use of virtual memory, it is beneficial to first investigate an abstract model that does not include an excessive amount of information. This is due to the fact that considering these different techniques might be quite complicated.

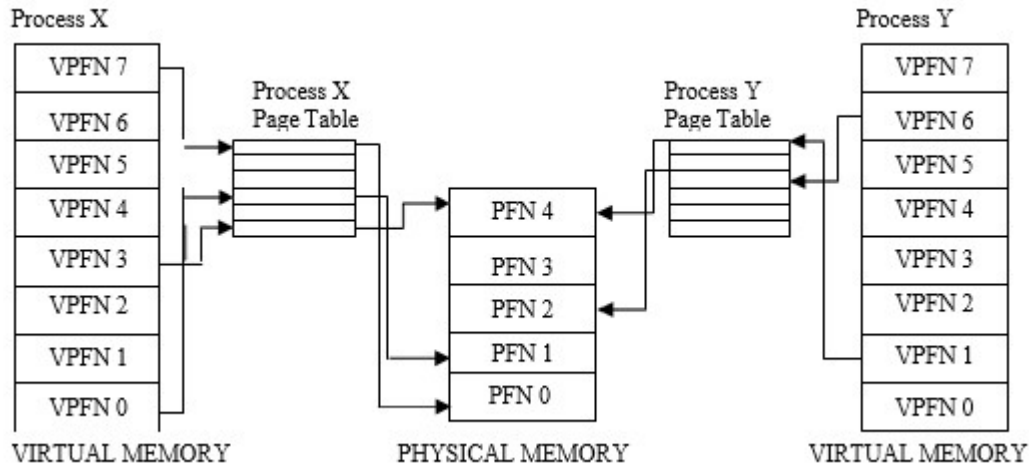


Figure 6.1: Abstract model of Virtual to Physical address mapping

source: fundamentals of operating system, data collection and processing through by Er. Nishit Mathur 2015

As part of the process of carrying out a program, the processor will access memory in order to get an instruction, after which it will decode the instruction. When decoding the instruction, it's possible that you'll need to get or store the contents of a place in the memory that contains operands. This might be required at any point during the process. The instruction is then carried out by the processor, and after that is complete, the processor moves on to the instruction that comes next in the program. This is one method that the central processing unit (CPU) uses to continually access memory so that it may either retrieve instructions or obtain and store data.

When utilized on a computer that has virtual memory, each and every one of these addresses is regarded to be a virtual address rather than a physical address. The information that is included in a number of tables that are controlled by the operating system is used to direct the processor in the process of translating these virtual addresses into their corresponding physical addresses. The processor is responsible for this conversion. In order to make the translation process easier, memory, both virtual and actual, is generally divided into more manageable pieces that are referred to as pages. These pages are all the same size as one another in terms of their dimensions. (It is not necessary that all the pages should be of same size but if they were not, the system would be very hard to administer). On computers that use the Alpha AXP

architecture, Linux makes use of pages that are 8 kilobytes in size, but on computers that use the Intel x86 architecture, Linux makes use of pages that are 4 kilobytes in size. In Figure 1, each of these pages has been given its very own unique number, which is referred to as the page frame number (PFN). This number identifies the page in relation to the frame.

In this paged architecture, a virtual address is composed of two distinct components: an offset and a number that corresponds to a virtual page frame. These components are separated by a colon. If a page is 4 kilobytes in size, the offset will be stored in bits 11 through 0 of the virtual address, and the virtual page frame number will be stored in bits 12 and higher of the address. When the processor encounters a virtual address, it is up to the processor to determine both the offset and the number of the virtual page frame. This is done whenever the processor encounters a virtual address. In order to access the location on the physical page that corresponds to the proper offset, it is the responsibility of the processor to first convert the virtual page frame number into the matching physical page frame number and then perform this conversion. In order for the CPU to accomplish this, page tables are utilized by the device.

The virtual address spaces of two processes, process X and process Y, each of which has its own page tables, are illustrated in Figure 1, which may be located at this location. The processes in question are X and Y. These page tables are responsible for converting the virtual pages that are utilized by each process into the appropriate physical pages that are stored within the memory. This indicates that the memory in the physical page frame number 1 is mapped into the virtual page frame number 0 belonging to process X, and that the memory in the virtual page frame number 1 belonging to process Y is mapped into the physical page frame number 4, respectively. In each individual entry in the table of theoretical pages, the following information may be found:

- The Valid flag indicates if the currently selected item in the page table is valid or not.
- The number of the actual page that is being described by this entry is referred to as the "physical page frame number," which is abbreviated as "PFN."
- **Information relevant to the administration of access controls:** This is an explanation of the numerous applications that may be found for the page. Is there a method for me to send it a message? Does it include executable code that can be utilized?

The number of the virtual page frame that is now being shown is utilized as an offset in order to get access to the page table. The number 0 points to the first element in the table, which would make the virtual page frame 5 the sixth element.

The processor needs to figure out the number of the page frame that corresponds to the virtual address, as well as the offset that is contained inside the virtual page, before it can convert a virtual address into a physical one. This is because the offset is part of the virtual page. If the page size is expanded to a power of two, masking and shifting may be achieved with relatively little effort on the user's part. If we look at Figure 1 once more and assume that a page has a size of 0x2000 bytes, which is equivalent to the decimal value 8192, and an address of 0x2194 in the virtual address space of process Y, then the processor would convert that address into an offset of 0x194 into virtual page frame number 1. In other words, a page would have a size of 0x2000 bytes and a decimal value of 8192.

The processor uses the number of the virtual page frame as an index into the process's page table so that it can get to its entry in the process's page table. This allows the processor to access its entry. If the item at that offset in the page table is legitimate, the processor will obtain the physical page frame number from this entry in the page table. If the entry is faulty, the process may have tried to access a part of its virtual memory that does not actually exist. This can happen if the entry is incorrect. Because the central processing unit is unable to determine the address in this circumstance, control must be transferred to the operating system so that it may make the appropriate adjustments.

The way in which the processor notifies the operating system of the fact that an appropriate process has attempted to access a virtual address for which there is no suitable translation is a mechanism that is specific to the processor and cannot be used by the operating system in any other context. In any event, it will be provided by the processor; this type of event is known as a page fault, and the operating system will be informed of both the faulting virtual address and the reason for the page fault. In order to repair a page defect, the following actions are carried out:

1. An enticement to the computer's operating system.
2. Save the registers, as well as the present state of the process that the application that is executing is using.

3. Figure out whether the trap was triggered by an error on a page and whether or not the reference to the page is appropriate.
4. If the previous question was answered in the yes, navigate to the appropriate page on the backing store and make a mental note of where it is located.
5. Find a frame that won't break the bank to utilize.
6. Read the required page from the backup storage into the frame that has become available. (During this I/O operation, the CPU could be scheduled to perform some other task).
7. Once I/O has completed, restore the registers and the process state of the process that was responsible for the page fault, and maintain the state of the process that is now being carried out. Do this while preserving the state of the process that is being carried out.
8. Modify the associated PT item so that it reflects the fact that the page that was just duplicated is now present in memory. This should be done by removing the reference to the page that was just copied.
9. Carry on with the instruction that was currently being carried out at the time when the page fault occurred.

In the case that this specific entry in the page table is correct, the processor will calculate the address of the beginning of the page in the physical memory by taking the number of the physical page frame and multiplying it by the size of the page. In other words, the processor will get the address at which the page begins. The next step, but certainly not the least important one, is for the processor to add the required offset to the command or data that it needs.

Continuing with the previous example, the first virtual page frame of process Y is mapped to the fourth physical page frame of the process, which begins at the location 0x8000 (4 times 0x2000). This allows the first virtual page frame to correspond to the fourth physical page frame. Following the addition of the byte offset value of 0x194, we have arrived at the conclusion that the total physical address should be 0x8194. Because of the way that virtual addresses are converted into physical addresses, the virtual memory can be mapped onto the system's physical pages in any order at all.

This is because of the technique by which virtual addresses are converted into physical addresses.

- **Page Replacement Policies:**

It is imperative that serious consideration be given to the concept of demand paging since it is important to the operation of virtual memory. This suggests that the operating system, and not the programmer, is responsible for directing the process of moving pages into and out of main memory in accordance with the requirements of the processes that are now executing. When a process calls for a resident page that is not immediately accessible, it is up to the operating system to decide which of the resident pages that are now available should stand in for the non-resident page. This determination is made by a portion of the virtual memory that is referred to as the replacement policy, and it is this portion that is in charge of creating it.

The problem of figuring out which page should be removed may be addressed in a number of different ways, but the objective of each tactic is the same: to arrive at a policy that identifies the page as the one that will not be referenced once more for the longest amount of time. The difficulty of figuring out which page should be removed can be done in a number of different ways. The following is a comprehensive list of policies governing the replacement of pages.

6.5.1.1 First In First Out (FIFO)

First In First Out (FIFO) is the name of a replacement strategy that chooses the page that has been kept in memory for the longest to be the one that is updated first. This ensures that the most important information is always accessible.

- **The Strange Phenomenon of Belady:** In the event that all goes according to plan, the number of page faults should decrease even as the number of page frames increases. Nonetheless, in the case of FIFO, there are several circumstances in which this overarching principle will not be valid! This peculiar occurrence is referred to as Belady's Anomaly. It is important to keep in mind that OPT's is not in any way impacted by Belady's strangeness at any point in time.

6.5.1.2 Second Chance (SC)

The policy known as First In First Out (FIFO) was given a little tweak in order to produce the policy known as Second Chance (SC). This was done in order to get around

the difficulty of having to replace a page that is often seen. This policy makes use of a reference bit, which is represented by the letter R, in order to keep track of websites that have been referred to in a reasonably recent manner. This bit receives the value 1 each time the page is referred to, therefore its status is always up to date. The operating system will, at certain intervals, cause all of the reference bits to take on the value 0 in order to discriminate between pages that have and have not been referred to in a relatively short amount of time. This is done in order to prevent pages from being referred to more than once.

The operating system is able to determine, with the assistance of this bit, whether or not prior pages are still being accessed (i.e., $R = 1$); in other words, whether or not $R = 1$. If this is the case, the page in question is moved to the very end of the list of pages, and its load time is changed so that it represents the fact that it has only very recently been stored into memory. This is done in order to ensure that the page loads as quickly as possible. The next step in the quest will be taken after that. As a result, pages that receive a high volume of traffic are given a "second chance."

6.5.1.3 Least Recently Used (LRU)

When it comes time to replace a page, the Least Recently Used, or LRU, replacement policy will make the decision to replace the page that has not been referred to for the longest length of time possible. The foundation of this method is the presumption that the recent past would, to a large extent, be analogous to the near future.

The operating system is able to keep track of when each page was viewed by either saving the time at which each page was referenced or by retaining a stack of references. The time at which each page was cited is stored in the operating system.

6.5.1.4 Optimal Algorithm (OPT)

The procedure of replacing the page with one that will remain unused for the greatest length of time possible is referred to as the "optimal algorithm," and it is described in further detail below. Even if it has the highest performance imaginable, we are unable to use it since we are unable to precisely predict how much time will pass in the future.

Consider with me an illustration of four different frames, as an example: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	4
	2	2	2	2	2
		3	3	3	3
			4	5	5

6.5.1.5 Least Frequently Used (LFU)

According to the Least Frequently Used (LFU) replacement policy, the frequency with which a page has been used in the past serves as the primary factor in determining whether or not that page should be replaced. This particular policy keeps a track of the number of times a certain page is loaded into memory. Pages in main memory that have higher counts are not overwritten even if they have a lower count than a page that is being replaced. This is because higher count pages are used more frequently than lower count pages.

6.6 THRASHING

The term "thrashing" refers to the behavior that might occur in a computer system when it spends more time correcting page faults than it does carrying out activities. Even though it is necessary to be able to manage page faults in order to make full use of virtual memory's benefits, thrashing still has a negative effect on the computer system. This is despite the fact that it is vital to be able to handle page faults. The higher the page fault rate, the larger the amount of transactions that the paging device has to perform in order to function properly. As the number of people waiting in line at the paging device increases, the length of time it takes to resolve a page problem will likewise increase. As a result of the transactions in the system waiting for the paging device, the usage of the CPU, the throughput of the system, and the reaction time of the system all decline, which results to a performance that is below what would be considered optimum for a system.

The greater the degree to which a system is capable of multiprogramming, the greater the possibility that it may thrash. The graph in Figure 4 is a visual depiction of the fact that there is an ideal degree of multitasking that enables the system to run at its greatest level of efficiency. This fact is illustrated by the fact that there is a sweet spot. After

reaching its maximum value, the percentage of time that the central processing unit (CPU) is being used will quickly begin to decrease again. This occurs when the capacity of the system grows beyond its limits as a result of a rise in the number of applications that are running concurrently on the system. This implies that it is extremely important to have a solid control on the load that is being placed on the system in order to prevent thrashing from occurring. It is essential to maintain the multiprogramming level that corresponds to the peak of the graph in order to ensure that the system that is shown by the graph will always work in the right manner. This may be accomplished by making sure that all of the nodes in the graph have the same amount of memory available to them.

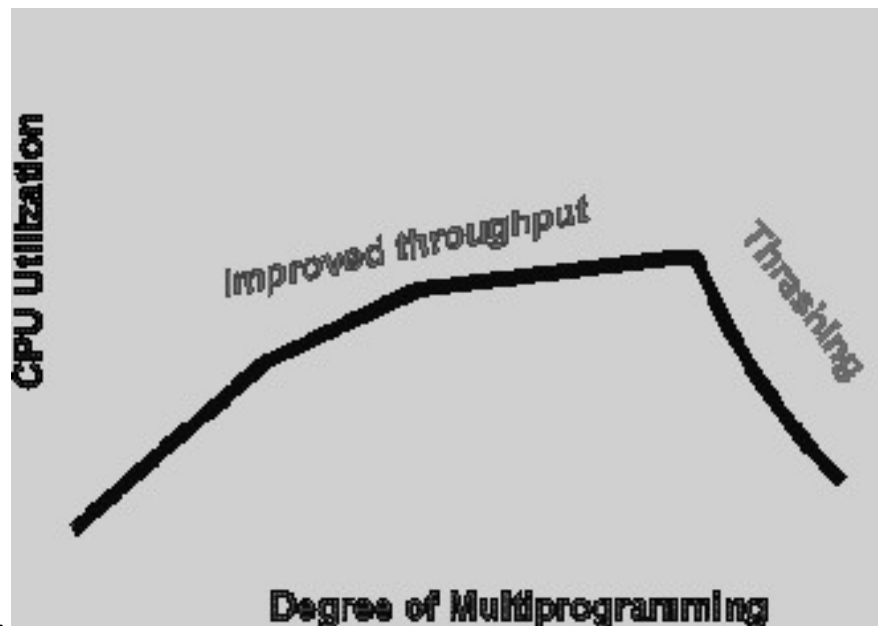


Figure 6.2 Degree of Multiprogramming

source: fundamentals of operating system, data collection and processing through by Er. Nishit Mathur 2015

The degree to which it will be feasible to eradicate the possibility of thrashing will be significantly influenced by the decision about which replacement approach will be used to set up virtual memory. There is a strong possibility that the effects of thrashing will be mitigated if a policy is put into place that takes into consideration the local mode. When the mode of operation is set to local, a transaction will replace pages from the

partition to which it is allocated whenever those pages are older than the transaction. Because transactions that use other partitions will not in any way be impacted by its requirement to access memory, there is no need to be concerned about that. If the partitions that are being used for the processing of other transactions contain an adequate number of page frames, then the processing of those other transactions will continue without interruption.

When a replacement technique is used that is based on the global mode, there is a greater chance that thrashing will take place. It is possible for a memory-intensive transaction to use up a significant amount of available memory, leaving subsequent transactions vulnerable to page faults and leading to a system that thrashes. This is as a result of the fact that each and every page of memory is accessible to each and every transaction. This is due to the fact that every single page of memory is available to every single transaction. In order to stop a process from thrashing, we have to guarantee that it has access to the exact number of frames that it needs. Both the Working-Set Model and the Page-Fault Rate are accessible options for strategies that may be used in order to solve this issue. Both of these strategies have the potential to be effective.

CHAPTER 7

I/O MANAGEMENT & DISK SCHEDULING

7.1 I/O DEVICES

Any piece of hardware that enables a human operator or other systems to interact with a computer is considered an input/output device. These devices are also commonly referred to as IO devices. As their name suggests, input/output devices are able to both send data to a computer (in the form of output) and take data away from a computer (in the form of input). A piece of hardware that can take in data, output data, or process data is referred to as an input/output (I/O) device. It takes in data as an input and transmits that data to a computer. Additionally, it takes data from the computer and writes it to storage media as an output.

7.1.1 Input System Components

The devices that are used to deliver signals to the computer so that it may carry out operations are referred to as input devices. The CPU, or central processing unit, acts as the last receiver and is tasked with the responsibility of transmitting signals to the various output devices. There are many different kinds of input devices, and some of the categories they fall into are as follows: keyboard devices; pointing devices; composite devices; game controllers; visual devices; and audio input devices.

7.1.2 The following is a description of some of the input devices.

Piano keyboard:

When it comes to feeding information into a computer, the input device that is utilized most frequently and to the greatest extent is the keyboard. The layout of the keys on the keyboard is comparable to that of a standard typewriter, despite the fact that there are some extra keys for performing various functions.

In general, keyboards are offered in two different sizes:

84 keys or 101/102 keys. However, at the moment, keyboards with 104 keys or 108 keys are also available for use with Windows and the internet.

- Different kinds of keys
- The Numeric Keyboard allows you to enter numerical data or move the pointer around the screen. The standard configuration has 17 individual keys.
- **Keyboard Shortcuts:** The number keys (0-9) and the letter keys (A-Z) are included in this group of keys.
- **The Command Sequence:** The pointer and the screen may both be controlled using these keys. It has directional arrow keys for four different directions. The control key, often known as Esc, is a combination of many other keys, including Home, End, Insert, Alternate (Alt), Delete, Control (Ctrl), and others.
- **Special Keys:** Some of the special function keys on the keyboard are Enter, Shift, Caps Lock, NumLk, Tab, and so on. Print Screen is also one of these keys.
- The Function Keys are located on the very top row of the keyboard, and their numbers range from F1 to F12.

The Mice

The mouse is by far the most prevalent type of pointing device. While clicking and dragging on the screen, a little cursor may be moved around the display with the use of a mouse. If you let off of the mouse, the cursor will become stationary. Your manual manipulation of the mouse is required for the computer to move in any direction; it will not do it on its own. As a consequence of this, it's considered an input device.

A mouse is a pointing device that allows the user to control the coordinates and movement of the cursor or pointer that appears on a computer screen by moving the mouse on a flat surface. It is possible to pick and move things using the left mouse button, while the right mouse button, when pressed, reveals additional menu options.

- **Trackball Joystick Mouse:** A pointing device that allows the user to move the cursor around the screen of a computer is called a joystick. Both the bottom and the top ends of the stick each have a ball of the same kind attached to them. The bottom spherical ball moves around inside of a socket. You have full control over the joystick and may move it in any of its four directions.

- **The joystick:** The functionality of the joystick is analogous to that of a computer mouse. CAD, which stands for computer-aided design, and playing video games on the computer are its two primary applications.
- **Ball on a Track:** The Track Ball is an alternative to using a mouse that may be added as an attachment to notebooks and laptops. Its anatomy is somewhat comparable to that of a mouse. The construction of it is similar to a ball with a hole cut in it, and we move the cursor with our fingers. Balls, buttons, and squares are some of the forms that may be employed for this purpose.
- **Bright Pointer:** A form of pointing device that has the appearance of a pen is referred to as a light pen. You may choose an item from the menu with it, or you can draw on the screen of the monitor with it. A photocell and an optical system are housed inside of a very small tube. When the button of a light pen is pressed and the tip of the pen is moved over a monitor screen, the photocell sensor element detects where on the screen the pen is being moved and sends a signal to the central processing unit (CPU).
- **Scanner in a Ballpoint Pen:** A photocopier and a scanner are both examples of input devices that perform comparable purposes. When there is information on study that has to be transferred to the hard disc of the computer so that it may be manipulated later, this method is utilized. The photos that are obtained from the source are gathered by the scanner, then converted to a digital format so that they may be stored on a disc. It is possible to make changes to these photographs before they are printed.

The scanner

- **OMR stands for optical mark reader:** A gadget known as an optical mark reader is typically utilized by educational establishments in order to validate the responses provided in objective examinations. It can distinguish between the markings made by a pencil and a pen.
- **OCR stands for optical character reader:** An optical character recognition reader (often known as simply an OCR reader) is a piece of hardware that can read printed text. OCR performs an optical scan of the text, converts it, character by character, into a code that can be read by a machine, and then stores the code in the memory of the system.

- **Magnetic Ink Card Reader (often abbreviated as MICR):** It is a gadget that is often utilized in banking institutions for the purpose of processing checks that have been provided by customers to the financial institution. It makes it easier to read the magnetic ink that is contained in the code number and the cheque number. When compared to other processes, this one moves along at a breakneck speed.
- **Readers of Bar Codes:** A device that can read information that has been bar-coded (information that is represented by light and dark lines) is called a bar code reader. Marking items with bar-coded data, such as number books and the like, is a popular practice. It might be a portable scanner or it could be a component of a fixed scanner. A bar code reader will receive an image of a bar code, then convert it to an alphabetic or numeric value, and finally transfer that information to the computer to which it is attached.
- **Internet Camera:** A webcam is considered an input device due to the fact that it captures a video image of whatever is happening in front of it. Either it is physically a part of the computer (as in the case of a laptop), or it is connected to the computer through a USB cable. A webcam is a very small digital video camera that is linked to a computer. Because it can both capture pictures and record video, it is also referred to as a web camera. The software needed to stream live video in real time over the internet is included with these cameras and must be installed on the user's computer before the cameras can be used. It is capable of shooting both still photographs and high-definition movies; however, the video quality is not as excellent as that of other cameras (whether in mobiles, other devices, or traditional cameras).
- **The Digitizer for Web Cameras:** A device known as a digitizer is one that is utilized in the process of converting analog signals to digital ones. It takes signals and turns them into numeric values. The Graphic Tablet is an example of a Digitizer. This device is used to transform graphical data into binary data.
- **The phrase "microphone":** The microphone serves as an input device that accepts speech signals and is also responsible for transforming them to digital form. It performs both of these functions simultaneously. It is a highly common component that can be found in just about every product that has anything to do with music.

- **Devices That Give Output:** After providing information to a computer system, the output of that information is shown on devices known as output devices. The output can take many various forms, including images, graphics, music, and video, among other possibilities. The following is a description of some of the output devices.

Monitors, also known as Visual Display Units (VDUs), are the primary output device used by a computer. It does it by organizing what are known as pixels, which are essentially microscopic dots, in a pattern that is rectangular. The number of pixels in an image is directly proportional to how sharp it is.

The following paragraphs will compare and contrast the two primary varieties of viewing screens that may be found in monitors.

- **Cathode-Ray Tube Monitor** (often abbreviated as CRT): Particles known as pixels are the fundamental building blocks of a cathode ray tube (CRT) display. The image quality or resolution is better, the smaller the individual pixels get.
- **Flat-Panel Display Monitor:** A flat-panel display monitor is a form of video display that, in compared to the CRT, has a smaller footprint, is lighter, and uses less electricity. You may either hang them on the wall or wear them on your wrist.

Calculators, video games, monitors, laptop computers, and graphical displays are just some of the products that now make use of flat-panel displays.

Monitor Television: The television is one of the most widespread output devices, and it can be found in virtually every home nowadays. As the user manipulates the television, it displays video and audio files on the screen. As a replacement for the CRT panels that were used in the past, we have transitioned to adopting plasma displays in recent years.

Printers are output devices that enable you to print information on study and are referred to as printers. The following is a list that describes the many kinds of printers available.

Printer Impact Printer, Non-Impact Printers, Character Printers, Line Printers, Laser Printers, Inkjet Printers etc.

In impact printers, the characters are printed on the ribbon, which is then pressed against the study until it is completely flattened. The following is a list of characteristics of printers that use impact technology:

- An extremely low overall cost for the consumables.
- Relatively loud
- Because of how inexpensive it is, it is an excellent choice for printing on a wide scale.
- There must be direct, physical interaction with the study in order to generate a picture.

Printers That Produce Characters: Character Printer is able to print a single character at a time and cannot print several characters simultaneously. It comes in two different flavors.

- Dot Matrix Printer
- Daisy Wheel Related Terms:

Line Printing Devices: Line Printers are a special kind of printer that can only print one line at a time on its output. It comes in two different flavors.

- Printing on Drums
- Printing on Chains

Printers that don't use impact: In non-impact printers, the characters can be printed directly onto the page without the use of a ribbon. Page Printers are another name for these kind of printers due to the fact that they print an entire page at once. The following is a list of characteristics of printers that do not use impact:

- More quickly
- They do not produce a significant amount of noise.
- Highest possible standard

- It is compatible with a wide variety of fonts and character sizes.

Laser Printing Systems: Laser printers make use of laser beams in order to generate dots, which are then combined to form characters on the study.

Printing using Inkjets: Inkjet printers are a type of printer that prints documents through the use of a spray method. An inkjet printer may generate sheets of an exceptionally high grade. In addition to that, they provide color printing.

Persons who speak: After receiving a command from a computer, speakers are able to create sound in response to that instruction. Speakers typically come equipped with wireless technologies these days, including Bluetooth speakers.

Projector: Projectors are optical devices that have the ability to present visuals on both sorts of screens, including those that are moving and those that are stationary. Displaying images on a large screen is made easier as a result of this. Projectors are typically utilized in performance spaces such as theaters, auditoriums, etc.

Prepare a plan: A plotter is a device that assists in the creation of graphics or other pictures to produce an accurate representation. To utilize these devices, you are needed to have a graphic card installed in your computer. These are the stylus-like instruments that are used to assist in the generation of precise drawings on a computer.

Readers of Braille: Users who are visually impaired have a significant need for a gadget known as a Braille Reader. It enables those who have limited vision or no vision at all to recognize the data by sliding their fingertips over the gadget, making it easier for them to interpret. It is a very significant technology for blind people since it allows them the comfort of understanding letters, alphabets, and other such things, which assists them in their academic endeavors.

The video card: A computer's visual capabilities are provided by a component called a video card, which is inserted into the motherboard of the machine. It provides assistance in the modification of digital material in output devices. It is a useful tool that assists individuals in using several gadgets at the same time.

The Global Positioning System, abbreviated as GPS: Because it use satellite technology to track the geometrical positions of users, Global Positioning System (GPS) is able to provide users with assistance in determining the appropriate directions

to take. GPS provides precise results because it performs continuous computations along both the latitude and longitude axes. These days, every smart gadget comes equipped with its own built-in GPS.

Audio headphones: The use of headphones is quite similar to the use of a speaker, which is often done by a single person or is a device that can only be used by a single person and is not typically done in vast regions. One other name for these devices is headsets, and they produce sounds at a lower frequency.

The Computer's Input and Output Devices Taken Together: There are an incredible number of gadgets that have the capabilities of both input and output simultaneously. They are able to carry out both activities simultaneously, receiving data and delivering results. A few of these are discussed in the following paragraphs.

Flash Drive USB: One of the devices that can do both input and output actions is a USB Drive since it can aid in both the receipt of data from one device and the transmission of that data to other devices.

To the modem: When it comes to the transmission of data through telephonic lines, modems are one of the most crucial equipment that may be used.

CDs and DVDs both: The compact disc (CD) and the digital versatile disk (DVD) are the most popular types of media that are used to save data from one computer to another in a specific format and to transfer data to other devices that function as input devices for the computer.

Headset (n.): A speaker functions as an output device, whilst the microphone acts as an input device, and the headset has both of these components: a speaker and a microphone.

7.1.3 I/O Buffering

Operating systems often make use of a technique known as buffering, which is designed to increase the effectiveness of input/output (I/O) activities. This is achieved by storing the data in a large number of different memory locations. In computer operating systems, the operation of temporarily storing data in a memory location known as a buffer or cache is referred to as buffering. Once the data has been buffered, it is feasible to recover it more quickly than the data's original source would allow for. This is because the buffering process makes the data more accessible.

Within a computer system, data can be kept in a variety of forms and places, including on hard drives, magnetic tapes, optical discs, and network devices, to name a few of the more common ones. If a process needs to read or write data from one of these storage devices, then it needs to wait until the device acquires or saves the data. This wait time is required whenever the process needs to read or write data from one of these devices. In the case that the procedure needs to read or write data from one of these storage devices, the waiting time is necessary. This period of waiting might be quite a while, in especially for those devices that are either slow or have a substantial amount of latency.

The use of buffering is one potential solution to the problem at hand. The phrase "buffer" refers to the temporary storage space that is produced by buffering, as the name suggests. The data may be briefly stored in the buffer before being transmitted to or retrieved from the storage device. Alternatively, the data may be permanently saved in the buffer. This will result in a reduction in the number of access operations that are required, which, in turn, will lead to an increase in the system's performance as a whole. After the data buffer has been entirely overwritten with new information, it is then sent in a batch to the storage device.

The Reasons Behind the Practice of Buffering: The buffering that occurs in operating systems is mostly caused by the following three reasons, which are described below:

- The act of buffering creates synchronization between two devices that run at different processing rates, which is an important function of the method. For example, buffering is required when the speed of the data supplier, a hard drive, is much higher than the speed of the data acceptor, a printer.
- Implementing buffering is required if two devices have data block sizes that are not identical to one another. This is one of the circumstances in which buffering is required.
- Buffering is another need that must be satisfied in order to fulfill the need to provide copy semantics for application I/O operations.

There are several distinct types of buffering: The following is a list of the three distinct types of buffering techniques that are available in various operating systems:

1. A Single Layer of Buffering
2. Dual Layers of Buffering
3. There are three different kinds of buffering that may be used: linear, circular, and circular.

7.1.4 Disk Scheduling (FCFS, SCAN, C-SCAN, SSTF), RAID, Disk Cache.

Through a procedure known as disk scheduling, operating systems are tasked with the job of allocating time slots for I/O requests that are scheduled to be written to a disk. Scheduling I/O operations is another term for the procedure known as disk scheduling.

The Crucial Role That the Disk Scheduling Function Plays Within Operating Systems:

- Multiple I/O requests could be issued by independent processes, but the disk controller could only handle one I/O request at a time even if it desired to serve all of them. This limitation prevented the disk controller from being able to service all of them simultaneously. As a direct consequence of this, further I/O requests need to be added to the waiting list and scheduled in accordance with their status.

It is possible that the movement of the disk arm will be enhanced as a consequence of the fact that two or more requests may be distanced from one another.

It is essential that hard drives be available in the most efficient manner possible since they are one of the components of the computer that are among the slowest.

Definitions of the Most Crucial Terms in Relation to Disk Scheduling:

- **Seek Time:** The time it takes to find the disk arm to a specific track on the disk where the data is to be read or written is referred to as the seek time. This time is included in the total time it takes to read or write data. Since this is the case, the method of disk scheduling that results in an average seek time that is as short as it may possibly be is the best choice.
- **Rotational Latency:** The rotational latency of a disk refers to the amount of time it takes for a desired sector on the disk to rotate into a position where it can access the read and write heads. This time can vary depending on the kind

of disk being used. Consequently, the disk scheduling technique that results in the least amount of rotational lag is the preferable alternative. This is because less lag means better performance.

- **Transmit Time:** The amount of time required to successfully transmit the data is referred to as the transfer time. It is based not only on the speed at which the disk is rotating but also on the number of bytes that are being communicated at any given moment.
- The Disk Access Time is configured to "First Come, First Serve," which is an abbreviation for "First Come, First Serve."

The File-Cluster-Focused File System (FCFS) is the one of the Disk Scheduling Algorithms that is the easiest to understand and operate. The FCFS file system processes requests in the order in which they are received in the disk queue. This ensures that requests are processed as quickly as possible. Let's try to make sense of this with the help of an example, shall we?

Those Customers Who Get There First Will Get Their Orders Served. Take, for instance: Take into consideration the demands in the following order: (82, 170, 43, 140, 24, 16, 190)

The Read/Write head is at its default position at the number 50 at the moment.

Therefore, the entire overhead movement (the total distance traveled by the disk arm) is 642, which is the product of $(82-50)$ plus $(170-82)$ + $(170-43)$ plus $(140-24)$ plus $(24-16)$ plus $(190-16)$. In other words, this number represents the whole distance traveled by the disk arm.

The Advantages of Employing an FCFS: The First Come, First Serve policy has a number of advantages, some of which are outlined in the following paragraphs.

- We provide each query with an equal chance to be taken into consideration.
- There will be no postponements for an unending period of time.
- The shortcomings of the current system, known as FCFS

The First Come, First Serve approach has a number of shortcomings, some of which are outlined in the following list.

- Does not make any effort to reduce the amount of time that is spent looking for anything
- There is a possibility that the provided service may not be of the best potential quality

SSTF is an abbreviation that stands for "Shortest Seek Time First,"

The SSTF method, whose name stands for "Shortest Seek Time First," gives higher processing weight to requests that have average search times that are completed in the shortest amount of time. Therefore, the seek time of each request is computed in advance in the queue. After that, the requests are scheduled in accordance with the seek times that were computed for them, and the process continues until all of the requests have been processed. Because of this, the request that is situated in a position that is physically nearest to the disk arm will be dealt with first when it comes to processing. SSTF is unequivocally superior to FCFS owing to the fact that it reduces the normal reaction time and increases the quantity of data that can be processed by the system. Both of these benefits are directly attributable to the fact that SSTF was developed. Let's try to make sense of this with the help of an example, shall we?

Take, for instance: First, in descending order of the quickest seek time. Take into consideration the demands in the following order: (82, 170, 43, 140, 24, 16, 190). The Read/Write head is at its default position at the number 50 at the moment. Therefore, the total distance traveled by the disk arm throughout the full overhead movement is equal to 208, which is $(50-43)$ plus $(43-24)$ + $(24-16)$ plus $(82-16)$ plus $(140-82)$ plus $(170-140)$ plus $(190-170)$.

The Advantages of Placing the Quickest Seek Time at the Top of the Listing:

The following is a list of advantages that result from adopting the approach with the least search time initially.

- The times at which responses are given are generally taking longer.
- There has been a rise in the amount of throughput

The following are some of the drawbacks of using the method with the shortest seek time: The method known as "Shortest Seek Time First" is connected with a number of problems, some of which are detailed in the following list.

- The cost of calculating the seek time in advance should be taken into consideration.
- A request may be turned down if it has a longer seek time in relation to other requests that are now being processed if there are currently other requests being processed.
- The significant variation in response times that may be expected since the SSTF gives certain requests higher priority than others

SCAN:

During the course of the SCAN algorithm, the disk arm travels in a certain pattern so that it may respond appropriately to the requests that come in along its path. When it reaches the end of the disk, it reverses its movement so that it can continue to fulfill the requests that are sent in its direction. As a consequence of this, the algorithm in question is frequently referred to as an elevator algorithm since it may perform the duties of an elevator. The requests that arrive before the disk arm will not be compelled to wait, but those that arrive after it will be completed less often. This is because the requests that fall inside the halfway are more likely to be granted.

Take, for instance: SCAN is the Name of the Algorithm. Supposing there are 82, 170, 43, 140, 24, 16, and 190 requests that need to be fulfilled, let's pretend there are these numbers. In addition, it is stipulated that the disk arm shall progress "towards the larger value," but the Read/Write arm is now located at the value of 50. As a consequence of this, the total overhead movement, which is also referred to as the entire distance covered by the disk arm, may be calculated as follows: $= (199-50) + (199-16) = 332$.
The Values That Can Be Obtained Through Employing the SCAN Algorithm

The SCAN Algorithm has a variety of advantages, some of which are described in the following paragraphs.

- High rates of throughput

- Constantly reliable response times
- Very little variance in response times
- Times of responses that are about average

The Problems That Can Occur When Employing the SCAN Algorithm: The SCAN Algorithm is not without its flaws, some of which are described in the following paragraphs.

Prolonged amounts of time spent waiting for replies to requests for locations that the disk arm is responsible for C-SCAN just stopped by not long ago. As a part of the SCAN procedure, the disk arm will make a second pass along the same path that it had previously scanned before it flipped its orientation and began scanning in the other direction. Therefore, it is possible that there are an overwhelming number of outstanding requests at the other end, or there may be zero or very few pending requests at the region that has been scanned. Both of these possibilities are possible.

The CSCAN algorithm finds a solution to this problem by directing the disk arm, rather than forcing it to reverse its orientation, to go to the other end of the disk and start processing requests from that point. This ensures that the disk continues to function normally. This eliminates the risk of experiencing the issues that were discussed before. Therefore, the disk arm moves in a circular motion, and because this method is similarly equivalent to the SCAN algorithm, we refer to it as C-SCAN, which stands for circular scan. Because of this similarity, the disk arm moves in a circular manner.

Example of scanning using a cyclical pattern. Supposing there are 82, 170, 43, 140, 24, 16, and 190 requests that need to be fulfilled, let's pretend there are these numbers. In addition, it is stipulated that the disk arm shall progress "towards the larger value," but the Read/Write arm is now located at the value of 50. Because of this, the total overhead movement, which is also referred to as the total distance travelled by the disk arm, may be calculated as follows: $= (199-50) + (199-0) + (43-0) = 391$. The C-SCAN Algorithm Offers Numerous Advantages. The advantages that C-SCAN has to provide are outlined in the following list.

In compared to SCAN, it enables a greater degree of continuity in the total amount of time spent waiting.

The LOOK:

LOOK method is extremely similar to the SCAN disk scheduling method, with the exception that the disk arm, despite traveling to the end of the disk, travels only to the final request to be handled in front of the head, and then reverses its course from there alone. In other words, the LOOK LOOK Algorithm is very similar to the SCAN disk scheduling algorithm, but with one key difference. Between these two methods, this modification is the only major difference that can be found. As a consequence of this, it gets rid of the additional latency that was caused by a needless traverse all the way to the end of the disk.

Consider, for Example, the LOOK Algorithm.

Supposing there are 82, 170, 43, 140, 24, 16, and 190 requests that need to be fulfilled, let's pretend there are these numbers. In addition, it is stipulated that the disk arm shall progress "towards the larger value," but the Read/Write arm is now located at the value of 50.

Because of this, the total overhead movement, which is also referred to as the total distance traversed by the disk arm, may be calculated as follows: $= (190-50) + (190-16) = 314$

CHAPTER 8

SECURITY & PROTECTION

8.1 DESIGN PRINCIPLES OF SECURITY

One definition of a distributed system describes it as "a collection of independent computer systems that are geographically isolated from one another but are connected to one another by a centralized computer network that is outfitted with software that is designed specifically for distributed systems." These computer systems make up a distributed system. In addition to carrying out the responsibilities that have been assigned to them, the autonomous computers will communicate with one another by trading resources and files.

The many different security design elements that need be applied in a distributed system are broken down into the following categories in this overview:

- Security should not be an afterthought; rather, it should be something that is built into the system right from the start.
- The users should not be made aware of any security measures, and such measures should not interfere with the usage of the system.
- The system should be designed to protect the confidentiality, integrity, and availability of data.
- The system should be intended to withstand assaults from both within and outside the system.

The basics are as follows: In a system that is spread among several nodes, there are a total of eight distinct security design ideas. These are as follows:

1. **The principle of "least privilege," or its application:** Users should only be provided the permissions that are absolutely necessary for them to carry out the responsibilities that have been allocated to them, in accordance with the principle of least privilege, which is an approach to the design of secure systems. As a consequence of this, you could also come across people referring

to this concept as the principle of the lowest possible authority. It is one of the most essential notions that must be adhered to in order to create a safe system, and it is regularly stated as such in several publications. It helps to maintain the safety of sensitive data and systems by preventing unauthorized access to those locations, which adds to keeping those areas safer overall. If users are only provided the permissions necessary for them to carry out the responsibilities for which they are accountable, there is a reduced risk of users being able to access information or systems to which they should not have access. This is because the risk of users being able to access information or systems to which they should not have access is reduced. Implementing it is not always easy, particularly in large organizations that have to support a broad variety of user profiles. However, it is usually worth the effort. Formalization of this concept has been incorporated into the designs of two different models: the Trusted Computing Base (TCB) model and the Security Kernel model.

2. **The idea of an economy of mechanisms:** A system ought to be created in such a manner as to decrease the number of independent components (for example, processes, machines, nodes, and so on) that need to interact in order to carry out a specific job in accordance with the idea of economy of mechanism. This may be accomplished by reducing the number of processes, machines, nodes, and so on. This concept is frequently referred to as the "principle of least action," particularly in the context of some communities. It is important to keep the design of a security system as plain and uncomplicated as is practically possible. This guiding principle is built on the idea that the complexity of a security system has a direct correlation to the number of holes that are available for an adversary to exploit faults in the system. Therefore, it is crucial to maintain security systems in a manner that is as plain and uncomplicated as is humanly possible. This is necessary in order to reduce the attack surface and make it more difficult for attackers to locate and take advantage of holes. The principle of economy of mechanism also goes by these titles, in addition to the names of the principle of least privilege and the idea of parsimony.
3. **The Fail-Safe Defaults Principle, According to Which:** The security settings that are pre-configured to prevent unauthorized users from accessing or making use of resources are the ones that go by the name of fail-safe defaults. Every user should, by default, have the fewest privileges necessary for them to execute the duties connected with their job function. This should be the case

regardless of the type of job function the user has. Access to sensitive information should be restricted to just those persons who have a compelling reason to make use of it. Data can only be protected against access by unauthorized users through the use of encryption. There is no other alternative. It is absolutely necessary for computer systems to be constructed such that they can survive attacks from malevolent users. Evaluation of the security controls is something that has to be done on a regular basis in order to confirm that they are effective.

4. **The Concept of a Totally Integrated Mediation:** The guiding principles of security design have to be all-encompassing and take into account any and all potential vulnerabilities and threats. It should be included into the system's overarching design, and its implementation should be carried out in such a way as to have as little of an effect as possible on the system's overall performance and usability. On a regular basis, it should be reviewed and kept up to date with any necessary changes.
5. **The idea behind an open design:** The idea of open design is a notion of security design that works to make security systems more accessible to unauthorized users. According to the open-design principle, security systems should be constructed in such a way that they can be easily examined, evaluated, and altered by anybody who possesses the required skills and experience. This should make it possible for a wide variety of people to participate in the system's development. This ought should be made possible by the open design of the system. The objective of open design is to make it easier for experienced security professionals to find and remedy holes in the security of a system in order to improve the overall security of the system. This will allow for the system's security to be improved. Because the design is open, it is also simple for security researchers to audit systems and determine the amount of protection they offer. It is possible to put the open design concept into effect with the help of a wide variety of open source software and hardware security tools and technologies.
6. **The idea of separating powers and responsibilities:** A user should not be able to access all aspects of a system to which they have been given access, in accordance with the principle of separation of privileges, which states that this should be the case. The goal of this concept is to protect systems from being

accessed by unauthorized users and to prevent users from intentionally or inadvertently causing damage to system resources. This is accomplished by stopping users from wreaking havoc on the system in any way. Having rights that are distinct from one another enables a system to more readily control access to its resources and protect itself from accidental or unexpected damage. It is standard practice to put the idea of separating privileges into effect by dividing a system into multiple levels, with each level having its own distinct privileges. This is done in order to put the notion of separating privileges into practice. The principle that describes this is called the separation of privileges. This concept is an essential part of the design of the security system, and it must be taken into consideration anytime any system is being created.

7. **The idea of using the Least Common Mechanism:** According to the principle of the least common mechanism, security systems should be developed in such a manner that the number of mechanisms that are shared by all users is kept to an absolute minimum. In other words, the number of mechanisms that are universally applicable should be as low as possible. This guiding idea is crucial because sticking to it reduces the possibility that a security flaw would be exploited by more than one person at the same time, which is why it is necessary. By reducing the number of mechanisms that are shared by all users, the concept of the least common mechanism makes it less likely that a security flaw may be exploited by an adversary who has access to the accounts of more than one person. This is accomplished by reducing the total number of mechanisms that are shared by all users. The number of common mechanisms has been reduced in order to reach this goal. The principle of least common mechanism is another name that the idea of least privilege goes by in some communities.
8. **The Acceptability Principle From a Psychological Standpoint:** The extent to which users are prepared to accept and comply with the security measures that have been developed in a system is referred to as the psychological acceptability of security design principles (also known as "psychological acceptability of security"). The notion that security measures need to be developed in a manner that takes into consideration the psychological components that play a role in the decisions of users regarding whether or not to accept and comply with those precautions is the fundamental concept that drives the principle. The idea is crucial because it contributes to ensuring that

users' data and privacy are adequately protected by security measures. This is why the concept is so important. Because of this, the idea is extremely important. It has been demonstrated that the psychological acceptability of security design ideas is affected by factors like the perceived efficacy of the security measures, the perceived simplicity of usage, and the regarded risks of not utilizing the security measures. When it comes to the designing of both physical and digital security measures, it is crucial to bear in mind the notion of psychological acceptability of security design principles. This is because psychological acceptability of security design principles can affect how people react to certain security measures.

8.2 USER AUTHENTICATION

Authentication was discussed previously, and both messages and sessions were brought up during that conversation. But what about the people who really buy things? If a computer system is unable to authenticate a user, there is no use in making an effort to verify that a message came from that individual if the system cannot authenticate the user. As a direct consequence of this, user authentication poses a substantial obstacle for the safety of operating system. The capacity to identify the now running applications and processes is, in turn, reliant on the capacity to identify each individual user of the system. This means that the security system is dependent on the ability to identify the currently running applications and processes.

The majority of the time, the user will identify himself to the system. How can we check the legitimacy of a user's account and make sure they are who they say they are? Authenticating a user typically requires one or more of the following three factors: the user being in possession of something (such as a key or card), the user being knowledgeable of something (such as a user identifier and password), and/or an attribute of the user (such as a fingerprint, retina pattern, or signature). For example, a user may be required to have a key or card in order to authenticate themselves.

8.2.1 To be recalled are the codes.

The usage of a user's password to validate their identity is the user authentication method that is utilized the vast majority of the time. After the user has submitted their user ID or account name, the user will then be required to provide a password for their account. The system will make the assumption that the account being accessed is being

done so by the owner of that account if the password that was entered by the user and the password that was remembered by the system are the same. In the lack of alternative, more complete forms of security, the use of passwords as a means of securing objects contained within a computer system has become standard practice.

Depending on how you look at it, they are a one-of-a-kind instance of either keys or capabilities. For example, a password might be associated with each resource (such a file), and then that password may be used to secure that resource. Every time a request is made to make use of the resource, the password must be supplied since it is mandatory that it be done so. In the event that the user supplies the correct password, access will be granted. There is a chance that different passwords will be linked to different access rights for a variety of reasons.

There is a possibility that you may need three different passwords in order to view files, upload new files, and update existing files. In practice, a user just needs one password in order to have complete access across the vast majority of systems. This is because passwords are case-sensitive. In practice, having a lot of passwords makes a system less secure, despite the fact that having a lot of passwords would theoretically make a system more secure. This is because the traditional trade-off that needs to be made is between convenience and security. In situations in which security precautions result in an unpleasant experience, those precautions are often circumvented or avoided in some other way.

8.2.2 Concerning Passwords, There Are Vulnerabilities

Passwords are used virtually everywhere due to the fact that it is not difficult to interpret them or put them into practice. Unfortunately, as we are going to illustrate in the next section, passwords may frequently be guessed, accidentally divulged, sniffed, or fraudulently sent from an authorized user to an unauthorized one. These are just some of the ways that passwords can be compromised. There are typically two different approaches that may be utilized while attempting to guess a password. One technique is for the intruder—whether they are human or a piece of software—to know the user or to have knowledge about the user. People have a bad tendency of creating their passwords off of facts that are easy to guess, such as the names of their dogs or the names of their significant others.

Use of brute force, which entails trying enumeration, which refers to each and every combination of permitted password characters (letters, numerals, and punctuation on

some systems), is the alternate choice. This can be done until the password is found. When confronted with passwords consisting of only a few characters in length, this tactic proves to be quite efficient. For example, a password consisting of four decimal digits only provides 10,000 unique permutations from which to pick. If you were to guess the solution 5,000 times, on average, you would be successful at least once. If a piece of software had the capability to test a password every millisecond, it would only take it around five seconds to figure out a password that comprised of four digits if the password was being tested at that rate.

When systems allow for lengthier passwords that can contain any combination of uppercase and lowercase letters, numerals, and punctuation marks, it becomes more difficult to successfully enumerate the passwords. This is due to the fact that lengthier passwords are more difficult to decipher. Users are expected to make advantage of the enormous password space that is available to them and cannot, for example, restrict themselves to only using characters in lowercase. Discovering a user's password can be accomplished in a variety of methods, including by guessing the password, observing the user visually or electronically, or doing both of these things.

Shoulder surfing is the act of an attacker peering over the shoulder of a user when that user is logging in. Shoulder surfing is referred to as "shoulder surfing." Shoulder surfing gives an adversary the ability to easily obtain the password by viewing what is being typed on the keyboard. Alternately, a person may simply add a network monitor to a computer if that person has access to the same network that the machine is connected to. This gives the person the opportunity to observe all data that is being sent on the network at the same time (also known as sniffing), which includes user IDs and passwords. Encrypting the data stream that carries the password is a simple solution that may be used to fix this problem. Nevertheless, even a system like this bears the danger of having its credentials stolen or otherwise tampered with.

In the event that the passwords are kept in a file, for instance, that file may be replicated such that it can be inspected by someone who is not logged into the system. Or, picture a software that gets installed on the computer and then pretends to be a Trojan horse so that it may record every keystroke that is made before sending that information on to the program that requested it. If the password is written down and stored in an area where it may be read by another person or lost, then you have a more substantial concern with exposure. We shall see that this may encourage users to write down their passwords or to reuse the same password in numerous different locations, since certain

systems demand users to select passwords that are difficult to remember or that are lengthy. This is because some systems force users to choose passwords that are either long or difficult to remember.

As a direct result of this, the degree of security that is provided by such systems is significantly lower than the level of security that is provided by systems that permit users to select passwords that are easy to remember. Unauthorized transmission is the third and final type of compromised password, and it is caused by human nature. This sort of compromised password is known as the "password leak." Due to the fact that it is against the rules of the great majority of computer systems, users are not permitted to share accounts on such systems. This regulation's principal objective is to strengthen safety and security, however it may be implemented for accounting purposes on occasion.

Consider, for example, the situation in which a number of users use the same user ID, and a security flaw occurs as a direct result of the usage of that user ID. It is not possible to establish who was using the ID at the time that it was hacked, nor is it possible to determine if the user was a valid one. Neither of these things is viable. Because there is only one user connected with each user ID, any user may be directly questioned about their usage of the account; in addition, the user may notice anything weird about the account, which might lead them to discover the intrusion. Because there is only one user associated with each user ID, any user may be directly questioned about their usage of the account. This behavior can result in an unauthorized user who may be harmful having access to a system. People sometimes disregard the rules for account-sharing in order to help their friends or to get past accounting requirements. Users have the option of manually choosing their own passwords, or they can choose to have the system generate passwords for them.

Because the algorithm generates passwords that are more likely to be difficult to remember, users may opt to write down their passwords instead. However, as was said before, user-selected passwords are frequently simple to decipher (for instance, the user's name or the user's favored vehicle). Certain security systems do an analysis on a proposed password to assess how straightforward it is to decipher or guess before they will allow it to be used. On some websites, administrators may occasionally check the passwords of users and will send a notification to the user in question if the password is simple enough to be broken. If the administrator discovers that a user's password is too easy to crack, the administrator may change the user's password. Some computer

programs have a function that "ages" passwords, which prompts users to change their passphrases at regular intervals (for instance, once every three months). The fact that individuals are able to switch between two passwords in such an easy manner is another reason why this strategy does not provide perfect security.

The solution, which has been implemented on certain platforms, is to preserve a record of all of the prior passwords that have been used by each user. For instance, the system may retain a record of the previous N passwords and stop users from repeating those passwords by preventing them from being reused. There are a few more strategies that may be utilized in conjunction with these fundamental password procedures. One illustration of this would be the availability of a more frequent option to change one's password. The password is changed between sessions so that we may follow things through to their natural conclusion. It is necessary to select a new password at the end of each session (either automatically by the system or manually by the user), and that password must be used for the future session in order to keep the system secure. In this situation, a password can only be used once, and this is the case regardless of how well it was typed. The real user becomes aware of the security breach during the subsequent session when the genuine user attempts to use a password that is no longer valid. After that, steps can be taken to fix the flawed security measures that were implemented.

8.2.3 Encrypted Passwords Passwords That Are Encrypted

When using any of these many methods, you run into the problem of trying to keep the password a secret within the computer, which is a hurdle in and of itself. How is it that the system may maintain a password in a secure location while at the same time allowing its use for authentication when the user submits her password? The UNIX operating system makes use of encryption as a method of getting around the requirement that its password list be kept a secret in order to function properly. Every user has their own one-of-a-kind password to log in with. The program includes a function that, although being simple to compute, is extremely difficult to invert and, according to the people who designed the system, may even be impossible to do so. If I may put it another way, if you are provided with the value x , it is not difficult to compute the value $f(x)$ of the function. This is the fifteenth chapter and it is titled Security $f(x)$.

If, on the other hand, you already know the output of the function $f(x)$, it will not be possible for you to calculate x . With the assistance of this technology, each password

is transformed into an encrypted form. Hashed versions of passwords are the only ones that are stored on file. When a user inputs their password, it is first encrypted and then compared to a previously saved encrypted version of the password. Even if the encrypted password that has been saved is observed, it is impossible to decode it, which implies that the password itself cannot be discovered. Therefore, maintaining the confidentiality of the password file is not a prerequisite in any way. The term "function"), in most contexts, refers to a data-encryption technique that has been painstakingly created and scrutinized before being used. When this strategy is put into action, there will be a flaw in that it will no longer be possible for the system to maintain control over the passwords.

Even if the passwords are encrypted, anybody who has a copy of the password file can use rapid encryption techniques against it. This is true even if the passwords are encrypted. For instance, they may encrypt each word in a dictionary and then check the resulting encrypted text against the passwords. If the user chooses a password that is also a word that can be found in a dictionary, then the password may be cracked and the user's account can be accessed. It is possible that piecing together a comparison will take no more than a few hours on machines that are sufficiently fast, or even on clusters of computers that are insufficiently fast. In addition, given that UNIX-based systems make use of a well-known encryption mechanism, a cracker may store a cache of passwords that have previously been cracked into in the past. This is possible because UNIX-based systems employ the same encryption method. Encrypted password entries are recorded in a file that only the system administrator, sometimes known as the superuser, is able to view in more recent versions of UNIX.

Because of the fact that these programs operate with the setuid root permission, only they are allowed to read this file; other users are unable to do so. This allows the applications to check a provided password against the stored password. The process of encryption also makes use of something called a "salt," which is also frequently referred to as a recorded random number. The inclusion of the salt in the password assures that even if two plaintext passwords are the same, the ciphertexts that are generated will be unique from one another. This is the case even if the plaintext passwords are similar. Another drawback of the UNIX password procedures is the fact that many UNIX systems only regard the first eight characters of a password to be important. This is only one of several UNIX password limitations. As a result of this, it is of the highest significance for users to utilize the allotted password space as much as they possibly can.

On many different platforms, it is forbidden to use words from dictionaries as passwords; this enables users to avoid having their data encrypted using the dictionary method. When creating a password, it is a good idea to use the initial letter of each word in a phrase that is easy to keep in your head. This will make the password much easier to remember. You must include both uppercase and lowercase letters, and you must also add a number or a punctuation mark somewhere in there for good measure. For example, the text "My mother's name is Katherine" may result in the password "Mmn.isK!" The user can easily remember the password despite its complexity, which makes it secure from prying eyes.

8.2.4 Passwords Generated Instantaneously

Shoulder surfing and password sniffing are two security threats that may be mitigated by using linked passwords, which can be used by a system. When a session begins, the system selects a password pair at random and displays just one half of it to the user. The other half is hidden. It is up to the user to give the other half of the password that is required to complete the login. The user will be given a challenge, and in order to proceed, they will need to submit the suitable response that is suited for that task. It is possible to extend the application of this tactic such that it also involves the utilization of an algorithm as a password. For example, the method could be formulated as an integer function in some implementations.

The system selects an integer at random, and the result is shown to the person who is using the system. The user invokes the function, at which point it returns a response that is formatted to display the desired outcome. The system is also responsible for performing the function. If both of the results are the same, then permission to enter can be given. A user may input a password, and even if someone were to steal that password, they would be unable to use it again in any capacity even if they got their hands on it. This is due to the fact that algorithmic passwords cannot be reused in any circumstance. When it comes to this kind, both the user and the system are privy to information that the other lacks. Never once does the information pertaining to the secret make its way through a medium that would make its revelation possible.

Instead, the data is entered into the function together with a shared seed and the function's own input. A seed might be a deterministic number, or it could be a string of alphabetic and numeric characters. A seed will be presented by the computer as an authentication challenge for the user to complete. Both the secret and the seed are

entered as their respective inputs into the function denoted by the notation $f(\text{secret}, \text{seed})$. The computer stores the outcome of this process as the password, and it is necessary to use this password in order to access the system. The fact that the computer is aware of both the key and the seed enables it to carry out a computation that is identical to the one performed by a human.

After comparing the two sets of findings, the user is said to have passed the authentication process. When there is a later need to authenticate the user, a new seed is generated, and then the exact same operations are carried out as before. This time around, the password is different from the last time. A system that only needs a one-time password will have a password that is different for each individual usage of the system. Anyone who tries to reuse a password from one session in another will fail if they capture the password from the first session and try to use it again in the second session. One of the few methods available to defend against erroneous authentication as a result of password leak is to employ passwords that are only used once. There is a wide variety of flexibility in the manner in which one-time password systems can be implemented. Hardware calculators are applied in commercial applications like as the SecurID product. Examples of these kinds of applications include.

The vast majority of these calculators have the look of a credit card, a key-chain dangle, or a USB device. They all include a display, but the existence of a keypad is not required for them to function. Some individuals utilize the current time as the random seed when they generate a random number. In order to access the shared secret using some other devices, the user must first enter a personal identification number (also known as a PIN; this abbreviation is sometimes shortened) using the keypad. After that, the one-time password will be displayed on the screen. The use of a personal identification number (PIN) in conjunction with a one-time password generator is an illustration of a type of authentication known as two-factor authentication. Given the nature of the situation, it is necessary to have two different sorts of components.

The level of protection that is supplied by two-factor authentication is noticeably superior than that which is provided by authentication based on a single element alone. One further twist on the idea of one-time passwords is the utilization of a code book, sometimes referred to as a one-time pad, which is a collection of passwords that are only good for a single login. Using this tactic, each password on the list is used one at a time, in the order that it was written down, and once each one has been used, it is either crossed off or deleted. The S/Key system, which is extensively used, uses either

a software calculator or a code book that is based on these calculations as a source of one-time passwords. The code book is based on the calculations that are performed by the software calculator. It should go without saying that it is the user's duty to keep the code book secure.

8.2.5 The Field of Research Known As Biometrics

The usage of passwords is one method of authentication that may be replaced with another method, which is the practice of utilizing biometric measures as a replacement. Hand- or palm-readers are often applied in the process of securing physical access, such as admission into a data center. This can be done to read a user's identification information. The information that is being read from hand-reader pads is compared by these readers to the criteria that have been established in the past. The parameters could include things like a temperature map, the length of a finger, the breadth of a finger, and even line patterns, amongst other things. These devices are now too bulky and pricey to be used in the regular process of computer authentication because of how the procedure is often carried out. Fingerprint scanners have experienced improvements in both their accuracy and their cost-efficiency in recent years, which suggests that their use will certainly increase in the years to come.

These gadgets will scan the ridge patterns that are present on your finger, and the resulting data will be transformed into a string of numbers. They will ultimately be able to store a number of sequences that will allow them to adapt for the positioning of the user's finger on the reading pad in addition to other characteristics. This capability is expected to become available in the near future. The program might then scan a finger that is placed on the pad and compare the attributes of that finger with the sequences that have been recorded in order to determine whether or not the finger that is placed on the pad is the same finger that has been saved. Even if there are a few different people using the scanner at the same time, it is obviously able to differentiate between the saved profiles of the various persons.

In addition to having a password, including a user name and a fingerprint scan in the authentication process can result in a highly accurate two-factor authentication method. This can be accomplished by incorporating the authentication process. It is possible that the system will be extremely resistant to being replayed or faked if this information is encrypted while it is being transmitted. A multi-factor authentication system offers an even higher level of safety. Think about how secure authentication may be if it

requires a personal identification number (PIN), a fingerprint scan, and a USB device that has to be hooked into the system. This method of authentication is not any more inconvenient than using normal passwords; the only difference is that the user must place her finger on a pad and plug the USB into the system in order to use it. Aside from that, there is no other distinction between the two. However, it is essential to bear in mind that strong authentication by itself is not sufficient to verify the user's identity. This is something that must be kept in mind at all times. If the session is not encrypted, then the integrity of the session might be compromised even if the session has been authenticated.

8.3 ACCESS CONTROL LIST

Access-lists, often known as ACLs, are sets of rules that are developed in order to manage network traffic and reduce the number of assaults on networks. ACLs are put to use to filter traffic according to a set of rules that have been created for the network's incoming or outgoing traffic, respectively.

ACL attributes include:

1. The set of rules that have been specified is matched in a sequential fashion, which means that the matching process begins with the first line, then moves on to the second line, then the third line, and so on.
2. The packets are only compared with one another until the rule is satisfied. When a rule is found to fit a certain condition, all additional comparisons are skipped, and the rule in question is executed.
3. There is an implicit denial at the conclusion of every Access Control List (ACL), which means that the packet will be rejected if neither the condition nor the rule is met.

After the access list has been created, it must be applied to either the inbound or the outward traffic of the interface:

- **Inbound access lists:** When an access list is applied to inbound packets of the interface, the packets will first be processed according to the access list, and then they will be routed to the outgoing interface. This happens when an access list is applied.

- **Outgoing access lists:** When an access list is applied to outgoing packets of the interface, the packet will first be routed, and then it will be processed at the outbound interface. This happens whenever an access list is applied to outbound packets of the interface.

Access-lists may be divided into two primary categories, which are referred to respectively as:

1. **Standard Access-list:** This is the Access-list that is created just by utilizing the source IP address. These ACLs decide whether or not the complete protocol suite is allowed or denied. They do not differentiate between the various types of IP communication, such as TCP, UDP, HTTPS, and so on. The router will recognize it as a standard Access Control List (ACL) if you use the digits 1-99 or 1300-1999, and it will use the address you specify as the source IP address.
2. **Extended Access-list:** This is the type of ACL that takes into account the source IP, the destination IP, the source port, and the destination port. With these different kinds of ACLs, we are also able to specify which IP traffic should be permitted or forbidden. These utilize the range from 100 to 199 and from 2000 to 2699.

In addition to this, there are two distinct types of access lists:

1. A numbered access list is an access list that cannot be erased precisely once it has been formed. This means that if we wish to remove any rule from an access list, we are unable to do so in the case of a numbered access list since this is not allowed. If we attempt to delete a rule from the access list, then the deletion of the rule will result in the deletion of the whole access list. Both the regular and extended access lists can make use of the numbered access list.
2. **Named access list** - In this kind of access list, a name is given to an access list so that it may be uniquely identified. In contrast to numbered access lists, named access lists can really have their entries deleted. In the same way that numbered access lists may be used with both regular and extended access lists, so can these.
3. Rules for ACL - 1 The basic Access-list is often applied in close proximity to the destination (albeit this is not always the case).

4. The expanded Access-list is often implemented in close proximity to the source, but this is not always the case.
5. We are only allowed to apply one Access Control List (ACL) per interface, regardless of the protocol or direction of traffic; this means that each interface can have only one incoming and one outgoing ACL.
6. If we are utilizing numbered Access-lists, we will not be able to remove a rule from an existing Access-list. In the event that we try to delete a rule, the entire ACL will be deleted instead. If we are making use of named access lists, then we have the ability to remove a particular rule.
7. Because any new rule that is added to the access list will be placed at the bottom of the access list, the entire situation should be thoroughly analyzed before implementing the access lists.
8. Because there is always an implied denial at the very end of an access list, we need to make sure that our access list has at least one sentence that says "permit," or otherwise all of the traffic would be turned away.
9. Extended access lists and standard access lists are not allowed to share the same name. Access-lists, often commonly referred to as ACLs, are sets of rules that are constructed in order to regulate network traffic and decrease the amount of attacks on networks. These rules are typically abbreviated with the letter "A." ACLs are put to use to filter traffic in accordance with a set of rules that have been developed for the network's incoming or outgoing traffic, respectively, and these rules are then applied to the traffic that is being filtered.

ACL characteristics consist of the following:

1. The rules that have been defined are matched in a sequential manner, which means that the process of matching starts with the first line, then continues on to the second line, then the third line, and so on. This continues until all of the rules have been matched.
2. The comparison of the packets with one another continues only until the criterion has been met. When it is discovered that a rule fulfills the requirements of a certain condition, all further comparisons are bypassed, and the rule in question is carried out.

3. There is an implicit denial at the end of every Access Control List (ACL), which indicates that the packet will be refused if neither the condition nor the rule is satisfied. This is because the implicit denial is a part of every ACL.

Following the creation of the access list, it must be applied to either the incoming or outgoing traffic of the interface:

- **Incoming access lists** - When an access list is applied to the incoming packets of the interface, the inbound packets will first be processed according to the access list, and then they will be routed to the outgoing interface. If an access list is not applied to the inbound packets of the interface, the inbound packets will not be processed. When an access list is applied, this occurrence takes place.
- **Outgoing access lists:** When an access list is applied to the interface's outgoing packets, the packet will first be routed, and then it will be processed at the outbound interface. This happens whenever an access list is applied to the interface. When an access list is applied to the outward packets of the interface, this occurrence takes place.

Access-lists are able to be broken down into two major types, which are correspondingly known as:

1. **Access-list on a Standard Basis** Simply using the source IP address to build an access list results in the creation of this Access-list. These access control lists (ACLs) determine whether or not the whole protocol suite is permitted or prohibited. They don't make a distinction between the many forms of IP communication, such as TCP, UDP, HTTPS, and so on and so forth. If you use the digits 1-99 or 1300-1999, the router will identify it as a normal Access Control List (ACL), and it will use the address that you indicate as the source IP address. If you use the numerals 100-199, the router will not recognize it as an ACL.
2. **A more comprehensive access list.** This particular variety of ACL takes into consideration not only the source IP and the destination IP but also the source port and the destination port. We are also able to designate which IP traffic should be allowed and which should be denied using these various forms of

Access Control Lists (ACLs). These make use of the ranges between 100 and 199 and between 2000 and 2699 respectively.

On top of this, there are two separate varieties of access lists, which are as follows:

1. An access list is referred to be a numbered access list if, after it has been created, it cannot be deleted in its whole. This implies that if we desire to delete any rule from an access list, we are unable to do so in the case of a numbered access list since doing so is not permitted. This means that if we wish to remove any rule from an access list, we are unable to do so. If we make an effort to remove a rule from the access list, then the removal of that rule will ultimately result in the removal of the entire access list. The numbered access list can be used by both the normal access list and the extended access list simultaneously.
2. **Named access list** - In this type of access list, an access list is provided with a name in order to enable one-of-a-kind identification of the access list. In contrast to numerical access lists, named access lists really let their items to be removed from the list entirely. These may be used with normal access lists as well as extended access lists, just like numbered access lists can be used with both types of access lists. Rules for ACL - 1 The normal Access-list will often be applied in close proximity to the destination, but this is not always the case.
3. The extended Access-list is typically put into place in close proximity to the source, but this is not always the case.
4. We are only permitted to apply one Access Control List (ACL) per interface, regardless of the protocol being used or the direction in which data is flowing; this implies that each interface may only have one incoming and one outgoing ACL respectively.
5. If we are using numbered Access-lists, we will not be able to delete a rule from an existing Access-list. This is because numbered Access-lists cannot be edited. In the case that we attempt to delete a rule, the entire ACL will be removed instead of just the rule in question. If we are utilizing named access lists, then we have the capability to get rid of a certain rule.
6. Prior to putting access lists into place, the entire issue has to be carefully examined so that any new rules that are added to the access list may be placed at the bottom of the access list.

7. Because there is always a line that implies a denial at the very end of an access list, we need to make sure that our access list contains at least one sentence that says "permit," or otherwise all of the traffic will be denied entry.
8. It is forbidden for extended access lists and normal access lists to have the same name.

One of the advantages of using ACL is that it helps to enhance the overall performance of the network:

- Helps to ensure the security of the network by enabling the administrator to modify the access list according to the needs of the company and obstructing the connection of unwanted data packets to the network.
- Provides you with the ability to manage the flow of traffic by granting or denying access to it based on the needs of the network at any given moment.

ACL's benefits include an improvement in the overall performance of the network:

- Contributes to the protection of the network by enabling the administrator to tailor the access list to the organization's requirements and prevent undesirable data packets from connecting to the network.
- Gives you control over the flow of traffic by allowing or blocking it depending on what the network requires at the time.

CHAPTER 9

UNIX/LINUX OPERATING SYSTEM

9.1 DEVELOPMENT OF UNIX/LINUX

Linus Torvalds began laying the framework in 1991 for what would eventually become the computer operating system known as Linux. Linus Torvalds is credited with having created Linux. The kernel of the computer operating system that is now commonly known as "Linux" was the first thing that people meant to refer to when they said the phrase "Linux."

Users are not obliged to pay any form of monetary contribution in order to access the program in question because it is open-source. It is helpful in the creation of computer hardware and software, as well as in the manufacturing of video games, mainframes, and products for a variety of other industries. It is able to run a diverse range of client applications from a variety of various platforms.

AT&T is the firm that is credited with the creation of the multi-user access, portability, multi-tasking, and bug-fixing operating system known as Unix. AT&T is also the business that was responsible for the design of the operating system known as Linux. In the outset, the company was run as a sole proprietorship by Ken Thompson, a former employee of Bell Labs who went out on his own and started the business.

It took some time, but finally it became the operating system that the most people utilized, which resulted in its broad acceptance. Its applications might potentially run on a variety of devices, including workstations, personal computers, and web servers. Users are supplied with access to a wide variety of programs that may be used in business.

Linux and Unix are two kinds of operating systems that are examples of software that are frequently used in commercial and server contexts respectively. Both of these types of operating systems are instances of software that can be found online. There are certain parallels that can be drawn between the two, but there are also some key distinctions that need to be emphasized. These similarities and contrasts should both be brought to the reader's attention.

What are some of the most important distinctions that exist between Unix and Linux?

Differences	Linux	Unix
Origins	Linux was developed in the 1990s by Linus Torvalds as a free and open-source alternative to Unix.	Unix was developed in the 1970s at Bell Labs
Introduction	<u>Linux</u> is Open Source, and a large number of programmers work together online and contribute to its development.	<u>Unix</u> was developed by AT&T Labs, different commercial vendors, and non-profit organizations.
Licensing	Linux, on the other hand, is open-source software and can be used freely without any licensing fees.	Unix is a proprietary operating system, meaning that it requires a license to use.
Kernels	both have a similar design but are less complex than the Unix kernel.	both have a similar design but larger and more complex than the Linux kernel.
Availability	On the other hand, Linux is widely used on both enterprise and personal computers.	Unix is typically found on enterprise-level servers and workstations and is less commonly used on personal computers.
Community Support:	Linux has a large and active community of developers and users who contribute to its development and provide support.	While Unix also has a community, it is generally smaller and more focused on enterprise-level users.
Accessibility	It is an open-source operating system which is freely accessible to everyone.	It is an operating system which can only be utilized by its copywriters.
Bug fixing time	Threat recognition and solution is very fast because Linux is mainly community driven. So, if any Linux client poses any sort of threat, a team of qualified developers starts working to resolve this threat.	Unix clients require longer hold up time, to get the best possible bug-fixing, and a patch.
File system supports	File system supports – Ext2, Ext3, Ext4, Jfs, ReiserFS, Xfs, Btrfs, FAT, FAT32, NTFS	File system supports – jfs, gpfs, hfs, hfs+, ufs, xfs, zfs

Graphical User Interface	Linux provides two GUIs, <u>KDE</u> and <u>Gnome</u> . But there are many other options. For example, LXDE, Xfce, Unity, Mate, and so on.	Initially, Unix was a command-based OS, however later a GUI was created called Common Desktop Environment. Most distributions now ship with Gnome.
Use Cases	It is used everywhere from servers, PCs, smartphones, tablets to mainframes.	It is used on servers, workstations, and PCs.
Shell Compatibility	The default interface is <u>BASH</u> (Bourne Again Shell). Anybody can use Linux whether a home client, developer or a student.	It initially used Bourne shell. But it is also compatible with other GUIs. Developed mainly for servers, workstations, and mainframes.
Source Code Availability	The source is accessible to the general public.	The source is not accessible to the general public.
Hardware Compatibility	Originally developed for Intel's x86 hardware processors. It is available for more than twenty different types of CPU which also includes an ARM.	It is available on PA-RISC and Itanium machines.
Virus Threats	It has about 60-100 viruses listed to date.	It has about 85-120 viruses listed to date (rough estimate).
Operating System Versions	Some Linux versions are <u>Ubuntu</u> , <u>Debian</u> GNU, <u>Arch Linux</u> , etc.	Some Unix versions are SunOS, <u>Solaris</u> , SCO UNIX, <u>AIX</u> , <u>HP/UX</u> , ULTRIX, etc.

9.2 ROLE & FUNCTION OF KERNEL

The operating system is responsible for managing both the hardware and software of a computer by utilizing a component known as the kernel in each instance. In essence, it is in charge of the management of memory as well as temporal activities performed by the CPU. It is one of the core components that makes up an operating system. The kernel acts as a bridge between the data processing that is carried out by programs and the data processing that is carried out at the hardware level. This is accomplished through the use of inter-process communication and system calls.

The operating system's core software, known as the kernel, is the very first component to load into memory when the operating system is booted up. It will remain in that location until the operating system has been entirely turned off. It is responsible for a wide range of tasks, some of which include memory management, task management, and disk management, among others.

- The kernel is equipped with a process table that keeps a record of all processes that are currently active in the system.
- The items in the per-process region table correspond to entries in the process table. This is because the per-process region table is a part of the process table.

Whenever the kernel receives the 'exec' system call, it will load an executable file into memory at that time.

9.3 DIRECTORY STRUCTURE

A directory is a listing of all of the files that are connected to one another on a disk. This listing is called a directory. There is a chance that the directory will not save some, part, or all of the properties of the files within it. It is possible to partition a single hard drive into many separate sections of different sizes. This gives the user the ability to utilize any of the many different file systems that are supported by the many different operating systems. There are a few other names for the partitions, the most common of which being volumes and small drives.

It is necessary for each partition to have at least one directory that is able to create a list of all of the files that are located on the corresponding disk. An entry is saved in the directory for each and every file that is kept in the directory. This entry stores all of the information that is pertinent to the particular file that is being stored. Every directory makes it possible to perform a variety of operations on files that are considered to be standard, including the following:

1. The Process of Creating Files
2. Search for the file that pertains to the question.
3. Deletion of existing files
4. Altering the name of the document in question

5. Investigating the Content of Several Files
6. A list of the documents in the inventory

9.4 SYSTEM ADMINISTRATION

You are able to utilize commands to manage numerous elements of AIX®, such as the system shells and environments, the resources that are accessible to the system, the process of starting the system up, backing it up, and shutting it down.

An individual who is responsible for administering an operating system is often referred to as the system administrator in written material connected to UNIX. This individual has the responsibility of managing the operating system. Unfortunately, due to the ease with which they are performed, only a select few of the tasks carried out by a system administrator may actually be categorized as administrative in nature. System administrators are tasked with a large number of tasks, thus these guides, along with others like them, are published with the goal of supporting them in fulfilling those obligations.

This particular operating system provides its very own individualized kind of help for system management in order to simplify the system's operation while simultaneously improving its safety and reliability.

- **The several management interfaces for systems that are now available:** The SMIT interfaces, in addition to the more conventional command line control of computer systems, are available through the use of this particular operating system.
- **Information crucial to the software product at hand:** The Software Vital Product Data (SWVPD) database includes specialized data relating to software products and the numerous different installable options that are available for those software products.
- **Improvements to the underlying operating system:** Each fileset that is included in the package that contains the operating system has a collection of client deliverable files that are logically related to one another. The filesets that comprise the operating system package are divided apart. It is possible to do individual installs of each fileset as well as updates on those installations.

- **Starting the computer from a cold boot:** After the main operating system has finished loading, the computer will instantly begin a complex set of activities. Whenever situations are within normal control, these obligations are carried out automatically.
- **A duplicate of the operating system:** After you have begun using your system, the next thing you need to consider is establishing backups of the file systems, directories, and individual files. This should be done as soon as possible after you have begun using your system. If you have a backup of your file systems, you will be able to retrieve any lost data or file systems in the event that your hard disk fails to function properly. When it comes to storing data and creating backups, you have a number of different alternatives to choose from.
- **Turning off the power to the computer system:** By utilizing the shutdown command, the operating system may be brought to a complete halt in a manner that is both safest and most thorough.
- **The surrounding conditions of the system:** The environment of the system is generally made up of a collection of variables that either define or govern certain aspects of the method in which a process is carried out. This is because the environment affects the outcome of the process.
- **AIX usage metric data for the IBM License Metric Tool, often commonly referred to as SLM tags:** Use metric data is gathered by the IBM® License Metric Tool from the Software License Metric (SLM) tags that are generated by the AIX operating system. These tags serve as input for the IBM® License Metric Tool. The Software License Metric (SLM) tool is responsible for the generation of these tags. The usage metric data stores the information required to determine the total number of virtual central processing units (vCPUs) that are currently online. The abbreviation vCPU is used to refer to this piece of information.
- **Knowledgeable with the AIX Operating System Runtime:** AIX Runtime Expert makes it easier to gather, apply, and test the runtime environment for one or more AIX instances by streamlining a sequence of activities that may be carried out against a single consolidation. This operation can be performed on an unlimited number of AIX instances.

- **Instructions as well as methods for actual operations:** A command is an instruction that tells a computer to carry out a certain action, such as launching a program or carrying out a specific operation. A process is the name given to a piece of software or an instruction that is in the process of being carried out on a computer.
- **Getting a handle on the flow of the system:** Users are able to keep operating mission-critical programs owing to the management of system hangs, which also increases application availability. This was made possible by the improvement of application availability. The detection of a system hang alerts the system administrator of any possible difficulties, at which time the administrator is given the choice to either log in to the system as root or to restart the system in order to address the issue. The administrator is notified of any potential issues upon the discovery of a system hang.
- **Process management and administration:** The process is the entity that the operating system uses as the one that is responsible for exerting control over the consumption of system resources. This control is exercised by the operating system. Even though threads have the capacity to affect the total amount of processing time that is utilized, the vast majority of system management tools still need you to refer to the process in which a thread is operating rather than the thread itself. This is the case despite the fact that threads can influence the amount of processing time that is utilized.
- **Taking into account the system as a whole:** You have the ability to gather data on individual and group consumption of various system resources, which you can subsequently report using the accounting software for the system. This data may be collected both individually and collectively.
- **Controller of the System's Available Resources:** It is the responsibility of the System Resource Controller, often known as SRC, to provide the system manager and the programmer with a collection of commands and subroutines that make it easier for them to construct and manage subsystems. These commands and subroutines are designed to simplify the process.
- **Files that are associated with the computer's operating system:** All information that is input and output (I/O) is handled by the operating system

through the usage of files. This is done so that access to software and hardware may be standardized. Both reading from and writing to files can happen in either way for a given file.

- **The outermost layers of computer operating systems:** A "shell" is just the name given to the piece of software that allows you to log into an operating system.
- **Precautionary safeguards for functional systems:** The protection of the data that is stored on a computer system should be the first priority when it comes to taking security precautions for computers.
- **User environment:** Each and every login name has a completely unique environment that is only accessible through that login.
- **A tutorial on how to use the BSD operating system:** This appendix is geared for being read by system administrators who have previous experience working with either the 4.3 BSD UNIX or System V operating systems. These specifics offer light on both the differences and the similarities that exist between AIX and the systems that were previously stated.
- **Rerouting of inputs and outputs simultaneously:** Because it offers you with specific I/O commands and symbols, the AIX operating system enables you to alter the data that is being input into and produced from your system. This is referred to as the input and output (I/O) of data.
- **Restoring the AIX operating system kernel:** As in AIX 6.1, the kernel has the power to recover voluntarily from errors in particular functions, therefore preventing an unexpected outage of the system. This functionality was introduced in AIX 6.1.

9.5 CASE STUDY

9.5.1 Linux

Linux is an operating system, much like Mac OS X from Apple, Windows from Microsoft, and iOS from Apple. In point of fact, Android, which is currently one of the most extensively used systems on the face of the planet, is powered by the Linux

operating system, which was mentioned before. An operating system is a piece of software that is installed on a desktop or laptop computer and is tasked with the management of all of the hardware resources that are attached to the device. To put this another way, the operating system on your computer is the one responsible for handling the communication that takes place between the software and the hardware. The operating system (OS) is necessary for the program to be able to carry out its functions; without it, the application cannot work.

The Linux operating system is made up of a number of different parts, including the following:

1. The bootloader is a piece of software that determines how your computer boots up for the very first time. When the majority of users turn on their computers, they won't see much more than a quick splash screen before it disappears and allows the operating system to load. This screen shows momentarily before going away to make room for the operating system to run.
2. **The Linux Core Operating System:** This is the only part of the operating system that can properly be referred to as "Linux." Because it is responsible for controlling the central processing unit (CPU), memory, and all of the other components of the system, the kernel is the most essential component of the operating system. An operating system cannot function without its core, often known as the kernel.
3. The init system is a type of subsystem that is accountable for the management of daemons as well as the initialization of the user space. In addition to this, it is responsible for the startup of the user space. Systemd, which is both one of the init systems that is used the most frequently and one of the init systems that has caused the most controversy, is also one of the init systems that is used the most regularly. After the first booting has been done by the bootloader, which is also known as GRUB or GRand Unified Bootloader, control of the boot process is passed over to the init system, which is responsible for managing the boot process and is in charge of controlling the boot process after the initial booting has been completed by the bootloader.
4. Daemons are background services (printing, sound, scheduling, etc.) that either start up when the computer boots or after you have logged into the desktop.

Daemons can either start up when the computer starts or after you have logged into the desktop. Daemons can either start up when the computer boots up or after you log in to the operating system.

5. The component of the system known as the graphical server is the one that is accountable for displaying the images on the screen of the monitor. The majority of people refer to it as the X server, however it may also be referred to simply as X.
6. The component of the program that the end users actually interact with is called the desktop environment. There is a vast variety of desktop environments to choose from, some of which are GNOME, Cinnamon, Mate, Pantheon, Enlightenment, KDE, and Xfce. Each desktop environment comes with its own assortment of applications already pre-installed on the computer. These applications might range from file managers and configuration tools to web browsers, games, and more.
7. **Different Functions and Applications** - Desktop environments do not provide access to the full suite of software programs. Linux, much like Windows and macOS, gives users access to tens of thousands, if not hundreds of thousands, of high-quality software titles that can be identified and installed with reasonable simplicity. This is true for each of these operating systems. The vast majority of today's Linux distributions (more on this topic will be covered later) have tools that work in a manner that is analogous to that of app stores and contribute to the centralized management and streamlining of the process of installing software. For example, Ubuntu Linux includes the Ubuntu Software Center, which is a rebranding of the GNOME Software application that enables you to search among the hundreds of apps that are available and install them from a single centralized place. GNOME Software was originally developed for Ubuntu Linux.

9.5.2 Windows Operating System

The graphical computer operating system known as Windows was initially developed by the corporation known as Microsoft. In addition to delivering these functionalities, it also enables users to view and save data, launch programs, participate in gaming and movie-watching, and establish a connection to the internet. It was made accessible for

use in both home computers and those utilized in professional environments. The very first version of Microsoft Windows was referred to as "version 1.0," and it was released in 1985. It was made accessible for use on home computers as well as in professional capacity under the Windows operating system on November 10th, 1983. After that, it became accessible for use in a wide variety of other versions of Windows, including the most recent edition, Windows 10.

1993 was the year when the general public was given access to Windows NT 3.1, which was the very first version of Windows designed primarily for use in a business setting. The business quickly followed up with the launch of Windows 3.5, Windows 4.0, and Windows 2000 in that order correspondingly. Windows XP from Microsoft In 2001, Microsoft made Windows available to the general public for the first time. Since then, the corporation has released several versions of the Windows operating system, each of which was designed with the goal of catering to either personal or business needs. The x86 standard hardware, such as that manufactured by Intel and AMD, was utilized as the basis for the design of this system. As a consequence of this, it is compatible with a wide variety of computer hardware, including that which is produced by HP, Dell, and Sony, in addition to personal computers that have been assembled at home by the user themselves.

The computer operating system known as Microsoft Windows was developed by the Microsoft Corporation, which is also the name of the company's flagship product. At this point in time, it is without a doubt one of the most well-known computer operating systems in the whole globe. It employs a graphical user interface, which is sometimes referred to as a GUI in some circles. Because of its capabilities, users will be able to do things like save data, watch movies, run programs, take part in games, and connect to the internet. On November 10, 1983, the first version of Microsoft Windows was made commercially available for purchase. This version, 1.0, was also the very first release of the Windows program. Windows XP, Windows Vista, Windows 95, Windows 7, 8, 10, 11, and 12 are only some of the versions of Microsoft Windows that are now available. Other versions include Windows 95 and Windows Vista.

In 1993, Microsoft released Windows NT 3.1, which was the first version of the Windows operating system designed particularly for business customers. Windows NT 3.1 was the first version of the Windows operating system. Windows 3.5, Windows 4/0, and Windows 2000 were the first versions of Windows to be made accessible to the public. Subsequent versions of Windows have since been made available. Before

Microsoft launched Windows XP in 2001, the corporation had previously developed several iterations of the operating system that were suitable for deployment in a variety of environments, including homes and commercial establishments. Conventional x86 hardware, which comprised CPUs manufactured by both AMD and Intel, was utilized in its construction. As a direct result of this, it was compatible with a broad array of hardware, including personal computers manufactured by HP, Dell, Sony, and other companies, among others.

CHAPTER 10

VIRTUALIZATION CONCEPTS

Utilizing a computer operating system, The concept of "virtualization" refers to a quality that may be found in computer operating systems. This quality is characterized by the fact that the system kernel makes it feasible for several different instances of partitioned user space to coexist. The procedure of installing virtualization software refers both to virtualization that is independent of the operating system and to the process of installing virtualization software. It is installed on top of an operating system that was previously installed, and we refer to the operating system that was already there as the host operating system.

Users take part in this kind of virtualization by installing the virtualization software into the operating system of their machine in the same way that they would install any other program. This is done in the same manner that they would install any other application. After then, the user makes use of this application to construct and operate a number of distinct virtual machines. In this instance, the virtualization software grants the user unfettered access to any of the built virtual machines that they may choose to utilize. Even if the virtualization program does not have access to the hardware driver, compatibility issues with the hardware may still arise when an operating system is virtualized. This is because the host operating system is the only one that can provide the essential support for hardware devices.

The capacity of virtualization software to convert IT resources that were previously hardware-based and required the use of specific software to operate into virtualized IT resources is one of the many benefits of using this software. Because the host operating system is a whole operating system in and of itself, the management and administration of the virtualized host may make use of the many OS-based services that are accessible as organizational management and administration tools. This is due to the fact that the host operating system is an entire operating system in and of itself.

Here is a list of some important services that are powered by operating systems:

1. Developing a backup strategy as well as a recovery plan.
2. Administration of the Safety Measures.

3. Inclusion into the Directory Services.

The following is a description of a number of important processes that are involved in virtualization that is based on an operating system:

1. The capabilities of the hardware, such as the network connection and the central processor unit, may be put to use.
2. Accessories that are attached to it and with which it may have a two-way conversation, such as a camera, a printer, a keyboard, or a scanner, for example.
3. Information that can be read from or written to, such as files, directories, and network shares; this information may be read from or written to.

The operating system may have the ability to give or revoke access to certain resources, depending on the nature of the program that is making the request and the user account that is presently logged in when the application is being executed. This ability may rely on the type of the program that is making the request and the user account that is now logged in. When an operating system makes the decision to hide certain resources, it indicates to a computer program that the results of its enumeration will not include them even if the program calculates those resources and then chooses to hide them. Despite this, the computer application has communicated with those resources, and the operating system has successfully managed an instance of interaction between the two. When viewed from the perspective of programming, this is how things appear to be.

Containerization and operating-system virtualization make it feasible to run programs inside of containers, to which just a part of the available resources are allocated. This allows for more efficient use of system resources. The fact that it is feasible to split resources is what makes this possibility a reality. After being executed within a container, a program that is intended to monitor the entirety of the computer is only able to access the resources that it has been granted, and it perceives those resources to be the only ones that are available to it. This is because the program is only able to view the resources that it has been given permission to view. Each and every operating system allows for the construction of several containers, which then each have the ability to get a percentage of the total resources that are accessible on the computer. Each container has the potential to hold a wide variety of software programs. It is not impossible for these programs to run independently of one another, in conjunction with one another, or even in parallel with one another.

10.1 FEATURES OF OPERATING SYSTEM-BASED VIRTUALIZATION

- **The compartmentalization of resources:** Operating system-based virtualization provides a high level of resource isolation, which enables each container to have its own set of resources, including CPU, memory, and I/O bandwidth. This is made possible because of the fact that each container may have its own set of resources. The fact that virtualization is carried out on an operating system makes it so that one may take use of this capability.
- Because they use the same operating system as their host, containers consume far less resources than traditional virtual machines do. As a consequence, this leads in a shorter amount of time required to set up the virtual machine, as well as a lower amount of resource usage.
- **Portability:** Since containers are extremely portable, it is easy to move them from one environment to another without having to make any modifications to the program that is running behind the container. This is made possible by the fact that containers are very portable.
- **Scalability:** Containers can be rapidly scaled up or down based on the requirements of the application, which enables applications to be exceptionally responsive to changes in demand. Scalability is enabled by the fact that containers can be easily scaled up or down. One of the most essential characteristics of containers is their ability to scale.
- Containers offer a high degree of security because they isolate the containerized program from the host operating system as well as from any other containers that may be executing on the same machine. This provides a level of protection that cannot be matched by other methods.
- Containers endure less overhead than traditional virtual machines do since they do not need to replicate an entire hardware environment. This frees up resources that would otherwise be used for this purpose. They are relieved of the burden of being required to do so as a result of this.
- **Simple Administration:** Containers are simple to manage due to the fact that they can be started, stopped, and monitored by following only a few

fundamental commands. This makes them very convenient. Because of this, they are incredibly simple to use.

A virtualization solution that is based on an operating system may give rise to extra requirements and worries associated with performance overhead, such as the following examples:

1. The operating system that is now being executed on the host device makes use of the central processing unit (CPU), the memory, and a great deal of other hardware and information technology resources.
2. The performance of the system as a whole is slowed down because calls to the hardware that originate from guest operating systems have to pass through many layers in order to get to and from the hardware.
3. Licence is normally required for the operating system that serves as the host, in addition to separate licensing for each of the guest operating systems that the host system is able to support.

10.2 ADVANTAGES OF OPERATING SYSTEM-BASED VIRTUALIZATION:

- **Increases in Resource Efficiency:** Operating-system-based virtualization provides increases in resource efficiency since containers do not have to simulate a whole hardware environment. This frees up a lot of computing power that would otherwise be wasted. Because of this, the resource overhead needed to operate the emulator is cut down significantly, which is a positive effect.
- **High Scalability:** Containers can be fast and easily scaled up or down depending on the demand, which makes it easy to adjust to changes in the workload. This is one of the benefits of the high scalability of containers. Because of this feature, it is easy to quickly adjust to changes in the total quantity of work that has to be completed. basic instructions: Because containers can be controlled using only a few basic instructions, they are very simple to administer. Because of this, deploying and maintaining a large number of containers at the same time is a simple process.
- Traditional virtual machines have a higher requirement for both resources and infrastructure, but operating system-based virtualization has a far lower

requirement for both of these items. As a consequence of this, traditional virtualization may frequently result in significant cost reductions.

- Containers may be deployed more quickly, which reduces the amount of time needed to run new applications or update old ones. This time savings is especially beneficial when dealing with large amounts of data.
- **Portability:** Because containers are incredibly portable, it is simple to move them from one environment to another without having to make any changes to the software that is operating inside of them. This eliminates the need for any kind of maintenance or troubleshooting.

10.3 DISADVANTAGES OF OPERATING SYSTEM-BASED VIRTUALIZATION:

- Protection Virtualization that is based on operating systems might potentially raise safety problems because containers utilize the same operating system as the host computer. This suggests that a security flaw in one container has the ability to affect all of the other containers that are operating on the same system at the same time. Moreover, this flaw may affect all of the containers simultaneously.
- There is a possibility that containers will not be able to provide perfect isolation between the various programs running inside them. This may result in a decrease in performance or a dispute about the availability of resources. Utilizing containers, on the other hand, does not create a totally isolated environment, which is one of the limitations of doing so.
- **Difficulty in both the Initial Setup and Day-to-Day Operation:** Virtualization that is based on an operating system can be difficult to set up and run, requiring the owner to possess a specific set of skills and a given level of knowledge. In order to do so, however, virtualization can be very useful.
- **A dependence on other people The issue(s):** There is a possibility that containers will experience dependency difficulties, either with the host operating system or with other containers. As a consequence of this, there is a risk that compatibility difficulties can occur, which will make it more difficult to install applications.

- Containers may have limited access to the resources of the underlying hardware, which may limit their capacity to carry out certain activities or programs that need direct access to the underlying hardware. This restriction may be imposed by the operating system or by the software that was used to create the container. Containers, for instance, might not be able to support the operation of virtual machines (VMs).

Would you be able to provide me an explanation of what virtualization is and how it works? It is possible that we will be able to arrive at a description of this concept that is more accurate if we enlist the assistance of the American corporation VMware. Therefore, virtualization is an abstraction layer that decouples the actual hardware from the operating system in order to provide more flexibility and improved utilization of information technology resources.

In the 1990s, researchers began to grasp how virtualization may be able to manage some of the challenges associated with the proliferation of hardware that was less expensive, growing costs of maintenance, and vulnerabilities. The goal of the first presentation of the notion, which took place in the 1960s, was to simplify the process of dividing large mainframe computers. Personal computers, on the other hand, made it possible to distribute computing power in a more efficient manner as time went on. In light of this, during the 1990s, academics started coming to the realization that virtualization may solve some of these issues.

A technique known as virtualization makes it feasible to run a number of distinct virtual machines, each of which may function independently on a single physical computer. This is made possible by the fact that it is possible to have more than one physical computer. Each of these virtual computers may be outfitted with a unique operating system of its own choosing. Each virtual machine has its own random access memory (RAM), central processing unit (CPU), network interface card (NIC), and disk space, all of which are loaded with an operating system and any programs that are currently being used. The operating system is only aware of a standardized and consistent collection of hardware, independent of the actual physical components that make up the hardware. This is because the collection of hardware has been standardized.

Because virtual machines are stored within files, it is much simpler to store, clone, and deploy a virtual machine in a short amount of time. This is yet another advantage of

using virtual machines, and it contributes to the ease with which they may be utilized. This function is known as "encapsulation of files," and that is the term that has been given to it. Full systems, which include applications, operating systems, BIOS, and virtual hardware that have been set up entirely, are able to be moved from one physical server to another in a matter of seconds for the purpose of performing maintenance with no downtime and continually consolidating workloads. This is possible due to the fact that full systems can be broken down into their component parts and moved independently of one another. It is possible to transfer whole systems from one physical server to another.

Users that engage in the process of virtualization are able to take advantage of a wide range of benefits, including the following:

1. The separation of the hardware from the software, which makes it possible for a single physical system to host a number of different applications and operating systems. In addition, the administration of computing resources as a single pool, which is subsequently distributed to managed virtual machines in a way that can be managed.
2. The separation that exists between the host computer and the virtual machines, as well as the functional autonomy of the virtual machines and the programs that are running on them; the only way for these virtual machines and programs to communicate with one another is through the use of network connections.
3. The ability to capture and save, in the form of a single file, the entirety of the environment of a virtual computer (this ability is also known as "Encapsulation").

The act of decoupling a resource or request for a service from the underlying physical delivery of that service is referred to as "virtualization," and the term itself refers to this process. In certain circles, virtualization is sometimes referred to as "cloud computing." The meaning of the term "virtualization" has been broken down into its component parts in this in-depth explanation. Computer programs are able to acquire access to more memory than is actually accessible on the device they are operating on by, for example, shifting data to disk storage in the background while the program is still running. This allows the programs to access more memory than is actually present on the device they are operating on. The utilization of virtual memory enables the

accomplishment of this goal. In a manner analogous to this, virtualization strategies may be implemented across several layers of an IT infrastructure, such as the networks, storage, hardware of a laptop or server, operating systems, and applications. There are several approaches to take in order to accomplish this goal.

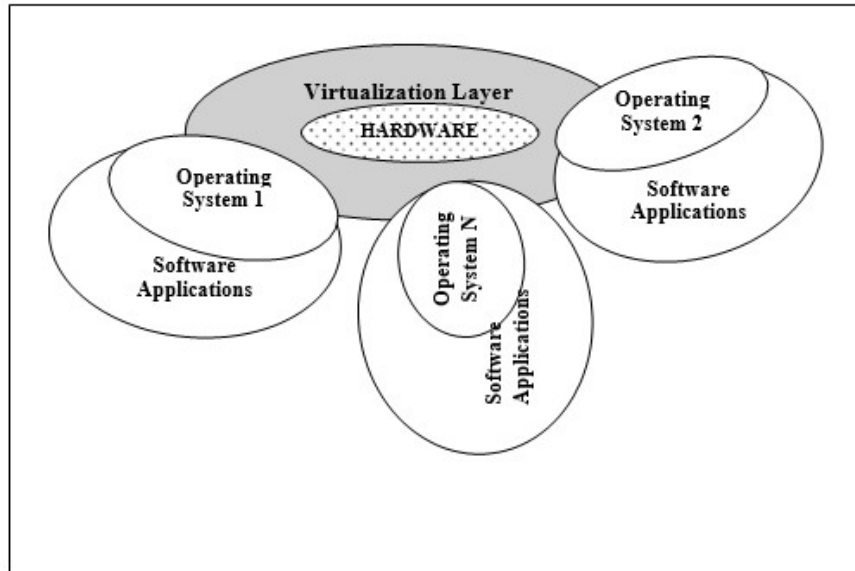


Figure 10.1: The Virtualization Layer provides a layer of abstraction between hardware and the applications

Source: *Virtualization concept, data collection and processing through by Lector dr. Logica Bănică by 2007*

This combination of virtualization technologies, which is frequently referred to as virtual infrastructure, adds a layer of abstraction between the hardware that is being used for computing, storage, and networking and the programs that are running on it (for more information, refer to Figure 1 of the attached figure). This layer of abstraction is useful for separating the hardware from the applications that are running on it. Because it does not significantly affect the experiences of the users of the system as a result of the implementation, the deployment of virtual infrastructure does not qualify as a technique that may be considered disruptive. On the other hand, managers who have access to a virtual infrastructure have the benefit of being able to more effectively manage shared corporate resources. This is one of the many advantages of having such an infrastructure. Because of this, IT managers are in a better position to adapt to the

ever-shifting requirements of their companies and to make better use of the infrastructure that is at their disposal.

The following information may be gleaned from a comparison of the functional structure of the computer before and after the virtualization process, if both of these states are examined side by side:

- As an alternative to a single operating system, the capability to run many operating systems on a single physical system while sharing the resources of the underlying hardware (partitioning).

Through the process of encapsulating whole systems into single files that are capable of being replicated and recovered on any destination server, virtualization makes high availability and error recovery solutions possible. o Virtualization makes it possible for operating systems and programs to function independently of the underlying hardware. o Virtualization provides high availability and error recovery solutions.

- Virtualization allows for hardware independence of operating systems and programs.
- Virtualization eliminates the need for unmanaged personal computers, workstations, or laptops.
- Virtualization layers a security policy in software that is applied to desktop virtual machines.

The company VMware was the first in the industry in 1998 to introduce the benefits of virtualization to computer systems that were based on the industry standard x86 architecture. These operating systems are now utilized in the vast majority of personal computers, including desktops, laptops, and servers. The word "virtualization" may be used to refer to a variety of various system levels, including the hardware level, the level of the operating system, and the level of the high-level language [2]. Some examples of these system levels include the hardware level, the level of the operating system, and the level of the high-level language.

The IBM mainframes that were available for purchase in the 1970s were the very first computers in the history of the world to ever include hardware-level virtualisation. Unix and RISC systems were the first to include capabilities for hardware-based

partitioning in more recent times. It wasn't until much later that these systems made the move to employing software-based partitioning, though.

Architectures that are capable of virtualization and can be used in the workplace:

When it comes to software-based partitioning, there are two ways [1] that are commonly used: hosted designs, and hypervisor designs. Both of these approaches are referred to as designs. These approaches are frequently used for industry-standard computer platforms like as x86 and Unix/RISC.

In addition to the operating system, a hosted solution (shown in Figure 2) offers partitioning services, making it compatible with the most varied assortment of hardware configurations. The capacity to function in the same way as an application while relying on the operating system of the host computer for support of devices and management of physical resources is the fundamental quality of virtualization. Virtualization was developed by IBM in the 1980s. On any given computer, there is a possibility that there are programs that are dependent on the host operating system as well as applications that are based on the Virtualization Layer. Both of these types of programs can coexist on the same device. Both varieties of software are compatible with running on the same computer.

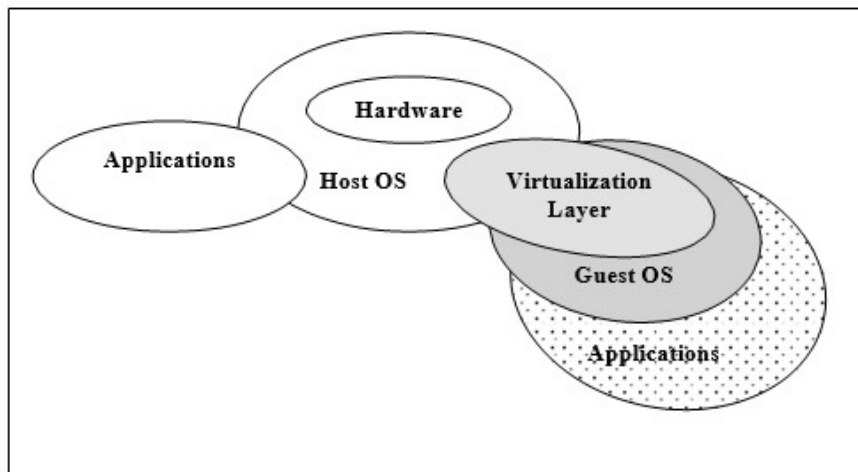


Figure 10.2: Hosted architecture

Source: Virtualization concept, data collection and processing through by Lector dr. Logica Bănică by 2007

When developing a hypervisor architecture, such as the one seen in Figure 3, virtualization is regarded to be the first layer of software that should be installed on a clean x86-based system. This is because virtualization allows a hypervisor to simulate many operating systems at once. The term "bare metal approach" is used to describe this procedure in some contexts. Because it can directly access the resources provided by the underlying hardware, a hypervisor is able to perform better than hosted architectures. This gives it a competitive advantage. This leads to improved scalability, robustness, and overall performance of the system.

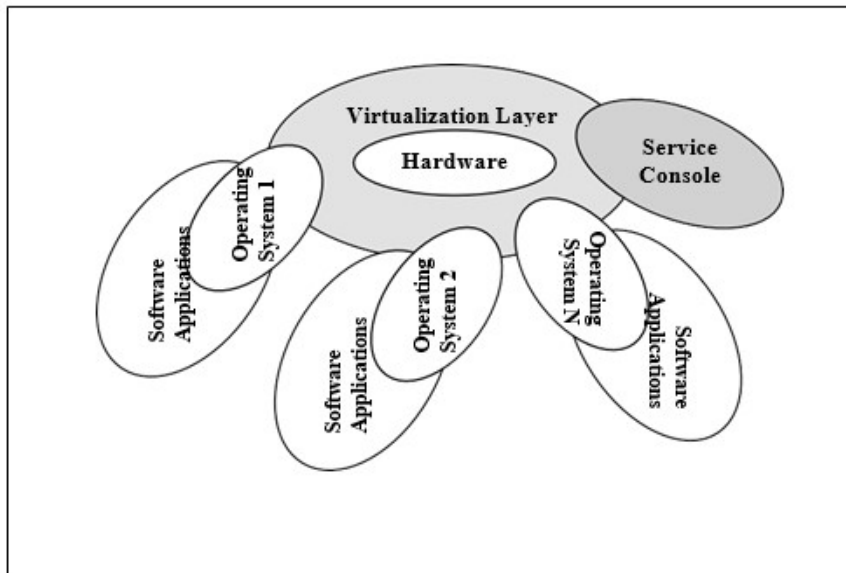


Figure 10.3: Hypervisor architecture

Source: *Virtualization concept, data collection and processing through by Lector dr. Logica Bănică by 2007*

In addition to a Service Console for application administration, this technique makes use of a central kernel for virtualization, which is referred to as the Virtualization Layer. This layer is also referred to as the Virtualization Layer. It is possible to program a hypervisor in such a way that it functions well in tandem with an operating system, or it is feasible to create it in such a way that it is not dependent on operating systems. The latter technique offers customers the ability to use an OS-neutral management paradigm, which ultimately leads to an increase in the data center's level of optimization.

One such option for application-level virtualization is called application-level partitioning. Using this approach, a number of different applications will use the same operating system. Although this approach provides less isolation than hardware or software partitioning (and has a higher risk), it does give limited support for older programs and environments that contain a mix of different types of software. In addition, this method only supports a small number of software applications.

The second design that is discussed in this chapter has fantastic performance since it provides advantages in addition to partitioning that are not available with the first architecture. It is possible to virtualize and manage a variety of system resources, such as CPUs, main memory, and I/O, and it also has the capacity of inter-partition resource management (service console).

Even while partitioning is a useful tool for IT organizations, the true value that genuine virtual infrastructure delivers to businesses extends much beyond that. Businesses stand to benefit in a number of other ways as well.

What are the steps I need to take in order to install various operating systems on the same computer?

In the increasingly informationalized corporate world of today, organizations of all sizes, from single proprietorships to global conglomerates, are exploring for less expensive means to expand their businesses with more complex and feature-rich software applications. These businesses are looking for ways to improve their operations using software that may save them money. They want access to the most latest technology, but they typically do not have the means or the need for a large number of servers that are capable of running several operating systems.

Through the use of server and workstation virtualization, they modify the way in which they manage the resources of information technology and give technological assistance to their business partners and clients. An excellent example of virtualization in action is provided by computer designs that support multiple processors. This is a reference to the technique of partitioning a single server such that it gives the appearance of being made of several servers. A single physical computer is now capable of running several operating systems and, as a result, a more wide and comprehensive collection of business applications thanks to the advent of virtualization software such as VMware Server and Microsoft Virtual Server.

At this point in time, it would appear that the most efficient approach to construct a virtualized information technology environment is to run software developed by VMware on equipment developed by HP. VMware was the first business to pioneer the virtualization of x86-based systems in 1998. The company continues to retain its position as the market leader in this field by providing the fundamental virtualization technology for all of the important x86-based hardware manufacturers. Using hosted (VMware Workstation and freeware versions of VMware Server and VMware Player products) and hypervisor (ESX Server) architectural designs, the company offers a comprehensive selection of software-based partitioning solutions.

The virtual machine (VM) approach that VMware takes provides a consistent representation of the hardware, which is subsequently implemented in software. This picture is then employed as the platform upon which operating systems and applications are performed in order to complete their respective tasks. On top of this platform, Virtual Center offers services for the management and provisioning of virtual machines, as well as the continuous consolidation of workloads across physical systems. These services are supplied in conjunction with one another.

The software for managing virtual infrastructure that is known as Virtual Center is what makes it possible for an organization's virtual machines to be controlled centrally and considered as a single, logical pool of resources. This is made possible by the fact that the program was developed. By utilizing Virtual Center, a system administrator is able to oversee and administer hundreds of Windows NT, Windows 2000, Windows 2003, Linux, and NetWare servers from a single point of management. The fact that Virtual Center is a centralized administration solution makes this accomplishment doable. The following computer operating systems are compatible with VMware: Windows 95/98/NT/2K/2003/XP, Linux (including Red Hat, SuSE, Mandrake, and Caldera), Novell (NetWare 4, 5, and 6), and Sun Solaris 9.

VMware was developed to provide compatibility with the existing software infrastructure investments that customers have already made. This was the motivation for VMware's creation. These expenditures do not merely entail operating systems; rather, they involve software for administration, high availability, replication, multi-pathing, and other services that are functionally equivalent. VMware was developed to assure compatibility with these investments and was named for the company that created it.

The Workstation software that is made available by VMware makes it simple to build and run several virtual machines on a single physical computer, regardless of whether that device is a stationary desktop or a mobile laptop. It is possible to convert a physical computer that is already being used into a virtual machine that is a representation of a whole computer, complete with RAM, network connections, and peripheral ports. This is a very useful capability.

Several different versions of Windows, Linux, and a wide range of other operating systems can be run concurrently on a single physical computer when a virtual machine is built using VMware Workstation. This functionality is available on all versions of Windows. The user have the capability to make exchanges.

You can switch operating systems instantly, access all of the PC's peripheral devices, and use a drag-and-drop feature to transfer data between virtual machines.

It is not possible for VMware Player to generate new virtual machines on its own; it can only handle guest virtual machines that were originally constructed by other VMware products. If you are interested in using it, you may get a free download of it. Users of VMware have unrestricted access to virtual disk images of a wide variety of pre-configured operating systems and applications. The vast majority of these virtual disk images were contributed by other VMware users.

There are more freeware software and websites (such as EasyVMX) that may be used to build virtual machines (VMs), mount VMware disks and floppies, manage VMware disks and floppies, and convert VMware disks and floppies. Some examples of these tools and websites are included below. Users of VMware Player are afforded the opportunity to construct, operate, and administer virtual machines at no cost (even when used for commercial purposes). Additionally, the VMware Player software is bundled together with the VMware Workstation installation. This was done so that it could be used for site-wide deployments with client users even if those client users did not have a license to utilize the whole VMware Workstation product.

Using VMware Server, one may generate virtual computers, edit them, and play back their creations. It utilizes a client-server architecture, which enables users to remotely access virtual machines. This allows for greater flexibility. Nevertheless, this does come at the cost of some of the graphical performance (and support for 3D, correspondingly). It is possible to run virtual machines that have been created by other

VMware products in addition to virtual machines that have been created by Microsoft Virtual PC. In addition to the capability of running virtual machines created by other VMware products, this functionality allows users to create their own virtual machines.

The hands-on component of the research focuses on the process of installing VMware Player, a free virtualization solution given by VMware Inc., on a Windows computer that is running Microsoft Windows Server 2003 SP2 as its primary operating system. This installation is carried out on a machine that is owned by VMware Inc. A specific driver is utilized by the Player in order to accomplish the goal of providing a distinct virtualization layer that is situated between the host operating system and the guest operating system. The image that may be found below depicts an installation of the open-source operating system known as React OS. An attempt is being made by the React OS project, which is based on opensource software, to recreate the Windows XP operating system family.

ISBN: 978-81-19534-89-0

Editors Details



Dr. Arun B Prasad is working as Principal at Dr V R Godhania College of Engineering And Technology, Porbandar, Gujarat. He has received his PhD degree in Computer Engineering from Pacific University, Udaipur in 2016. He has published 16 international research paper and have two patents in artificial intelligence. He is a keen observer, planner & implementer with track record of developing operational policies/ norms, systems & controls, motivational schemes & education standards for professionals during the career span. Prof Sanjay Agal is a renowned computer engineer and an esteemed author in the field of computer engineering and technology. With a lifelong passion for computers and technology, Prof Agal has dedicated her career to advancing the field and imparting her vast knowledge to aspiring engineers. Prof. Agal's journey in computer engineering began during her undergraduate studies, where she developed a deep fascination for the inner workings of computers and their transformative potential. Eager to explore the field further, she pursued a Master's degree in Computer Science, delving into topics such as programming languages, algorithms, and system design. Driven by her thirst for knowledge and a desire to make meaningful contributions, Dr Agal went on to earn her Ph.D. in Computer Engineering. His doctoral research focused on developing innovative solutions for improving the performance and efficiency of computer systems. His groundbreaking work in the field earned her recognition and accolades from leading academic institutions and industry experts. Throughout her career, Dr. Agal has held prominent positions in academia. As a professor of computer engineering, he has mentored and inspired countless students, sharing his expertise, and guiding them in their pursuit of knowledge. His teaching style is known for its clarity, precision, and ability to distil complex concepts into easily understandable principles. In addition to his academic endeavours, Dr. Agal has collaborated with industry leaders to implement cutting-edge technologies and address real-world challenges. His hands-on experience in the field has provided her with valuable insights into the practical applications of computer engineering principles.

Xoffencer International Publication
838- Laxmi Colony, Dabra,
Gwalior, Madhya Pradesh, 475110
www.xoffencerpublication.in

