Welcome

Operating System– An Introduction
by
Girish Godbole

Operating System – An Introduction

An Operating System (OS) lies in the category of system software.

An Operating System (OS) manages all the resources of the computer.

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

An Operating System (OS) acts as an interface between the software and different parts of the computer or hardware.

An Operating System (OS) is a fully integrated set of specialized programs that handle all the operations of the computer

An Operating System (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs

Operating System – An Introduction

Examples of Operating Systems

- **Windows** (GUI-based, PC)
- **GNU/Linux** (Personal, Workstations, ISP, File, and print server, Three-tier client/Server)
- **macOS** (Macintosh), used for Apple's personal computers and workstations (MacBook, iMac).
- **Android** (Google's Operating System for smartphones/tablets/smartwatches)
- **iOS** (Apple's OS for iPhone, iPad, and iPod Touch)

Why is operating system hardware dependent –

Operating system fundamentals do not depend on hardware directly. But hardware does influence certain aspects of the operating system like memory management and performance

How is it different from other application software -

An operating system is system software that acts as an interface between the user and the hardware, whereas application software is a program that performs a specific task. It is impossible to install the application software on a computer system without an operating system.

Operating System – An

Application Software	Operating System
It is a type of software that is created to do a certain set of tasks. It is a form of software that runs or executes on the user's request.	It acts as the interface between the user and the system hardware.
Application software can be written in different languages, including Java, Visual Basic, C, and C++.	Operating systems are typically written in C, C++, or Assembly.
It is available for buy and download on the internet. The installation packages were then used to complete the installation.	It is preinstalled in the system when the device is purchased.
When a specific task needs to be completed, the user opens this. The duration of the task determines its execution time.	It begins (boots) when the user turns on the computer and ends (shuts down) when he turns it off.
It is less important than the Operating system. It isn't possible to use it without a working operating system.	It's important since a computer can't operate without an operating system.
The primary goal of application software is to do a certain task. VLC Media Player, Picasa Photo Viewer, WhatsApp.	To manage hardware resources effectively. Windows, Linux, NOS, DOS, Unix, etc.

Operating System – An Introduction

The basic organization of a computer system is the

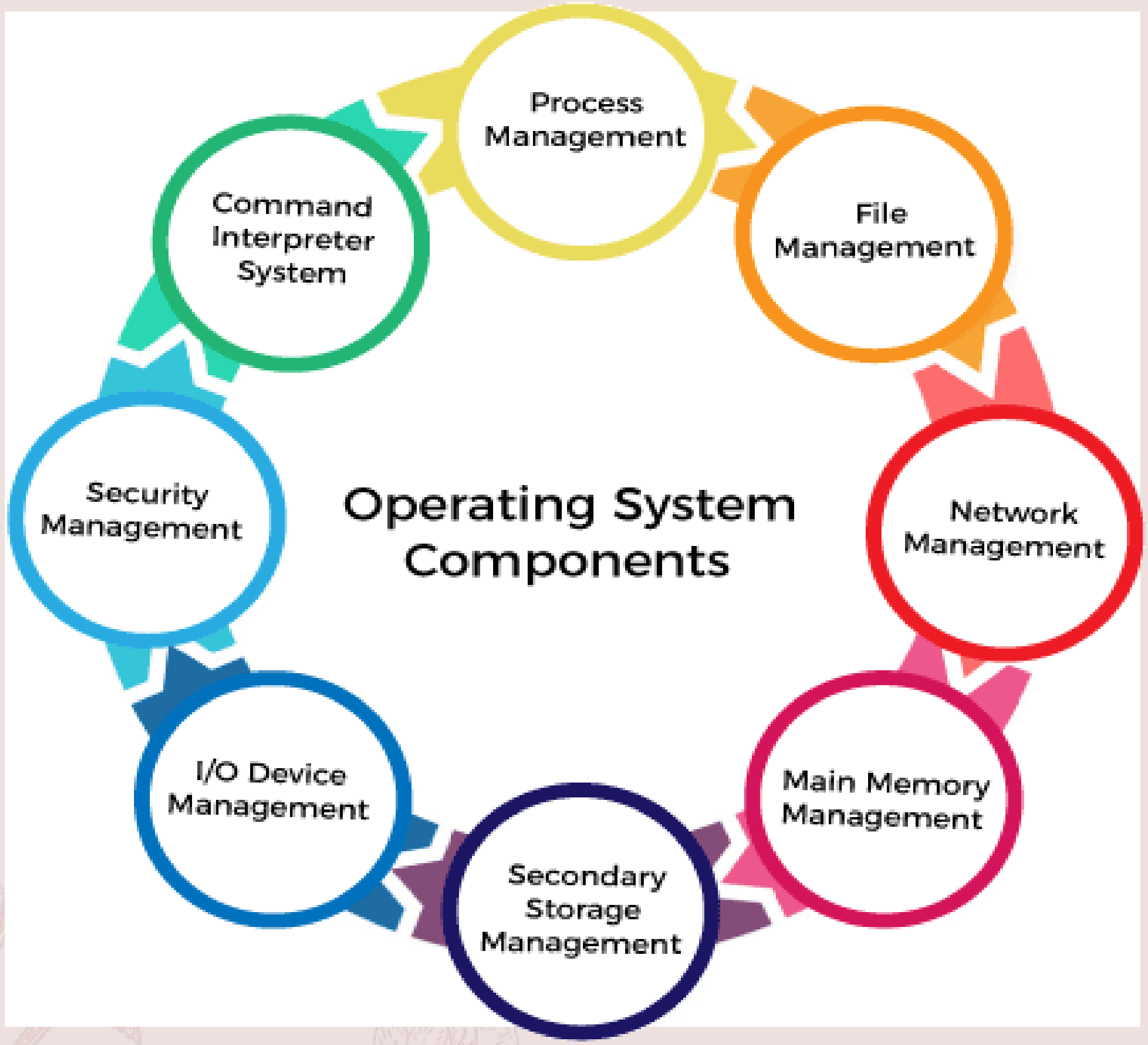
- Processing Unit,
- Memory Unit (arithmetic unit and the other is a logic unit)
- Input-Output Devices.

Why are Operating Systems Used?

Operating System is used as a communication channel between the Computer hardware and the user. It works as an intermediate between System Hardware and End-User. Operating System handles the following responsibilities:

- It controls all the computer resources.
- It provides valuable services to user programs.
- It coordinates the execution of user programs.
- It provides resources for user programs.
- It provides an interface (virtual machine) to the user.
- It hides the complexity of software.
- It supports multiple execution modes.
- It monitors the execution of user programs to prevent errors.

Operating System – Components



- Process Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- I/O Device Management
- Security Management
- Command Interpreter System

Operating System – Functions

Functions of the Operating System

- Process Management
- Memory Management
- Security
- Job Accounting
- File Management
- Device Management
- Networking
- User Interface
- Backup and Recovery
- Resource Management
- Virtualization
- Performance Monitoring
- Time-Sharing
- System Calls
- Error-detecting Aids

Operating System – Types

- Batch Operating System
- Multi-Programming System
- Multi-Processing System
- Multi-Tasking Operating System
- Time-Sharing Operating System
- Distributed Operating System
- Network Operating System
- Real-Time Operating System

Operating System – Types

Desktop Operating Systems
There are numerous types of operating systems software available for desktops. This section will help you understand each type and its unique features.

Windows Family
Windows has shaped the operating system industry with its continuous focus on innovation and constant improvement. The various types of window operating systems, which include versions such as Windows Vista, Windows 11, and more, are popular for critical features such as
Graphical User Interface
File Management
Window Management
Device Compatibility
Security Features
Microsoft Windows is one of the most popular types of operating systems for desktops worldwide. Windows currently retains a market share of over 74%.
Windows’s security, flexibility, and adaptability make it an ideal solution for various use cases. These include managing personal individual tasks as well as supporting large organizational workloads. Its compatibility with different types of hardware and software can also enhance productivity in business environments while promising scalability.

Operating System – Types

macOS

macOS, powered by Apple, stands out as a unique combination of functionality and user-centric design. It has robust security features, functionality, and continuous aesthetic improvement

Unique features and characteristics of macOS include:
Spotlight Search Tool
Intuitive User Interaction
Seamless Continuity
Data Backups

While macOS is celebrated for its elegant designs and interface, it is most known for its seamless integration with other solutions in the Apple ecosystem
The desktop operating system can easily be integrated with various Apple devices such as iPhones and Apple Watches. This can enable users to synchronize their data, efficiently share information more, and seamlessly transition between devices.
This integration benefits users with numerous Apple devices, as it can intuitively understand and address data requirements and connectivity preferences.

Operating System – Types

Linux Distributions
Linux is known as one of the most diverse types of networking operating systems. They offer numerous types of solutions, such as Ubuntu, CentOS, and much more. These different distributions each contain a unique combination of various Linux tools and capabilities, with each designed to address individual preferences and requirements.
For example, Ubuntu is best suited for users prioritizing adaptable interfaces and regular updates. In contrast, CentOS is popular for its robust tools and stability. Other benefits of Linux distributions include
Customization
Robust security mechanism
Server Capabilities
Long-Term Support(LTS)

One of the biggest advantages of Linux distribution is its intrinsic open-source nature and extensive community support channels. This allows users to inspect, modify, and distribute underlying source code freely. This fosters a culture of shared knowledge and encourages active participation in the development process.
This continuous focus on user-driven improvement helps facilitate a never-ending advancement cycle. This has led to a collective commitment to shared growth and innovation.

Operating System – Types

Mobile Operating Systems

Android

Android is one of the most popular and reliable types of operating systems for mobile devices. One of the most significant advantages of the Android operating system is its wide range of device compatibility and opportunities for customization. Users can use Android OS to personalize and tailor their experiences.

There are also numerous versions and variations of the Android operating system available. While this allows users to select the version that best suits their preferences and devices, it can also lead to fragmentation. This issue arises from the coexistence of multiple versions of the same operating system across different devices. Fragmentation can lead to inconsistency in user experiences and potentially staggered updates.

Key benefits and features of these different types of operating systems include

- **Application Ecosystem | Integration With Google Solutions**
- **Notification Management | Digital Wellbeing**

Android's appeal is most visible in its vast amount of device compatibility and opportunities for customization. Users can tailor these types of operating systems to suit their personal preferences. For example, Android operating systems can allow you to adjust home screen layouts, sideload applications, and much more. This level of adaptability is empowered by its active developer community that continuously drives innovation and user-centric evolution

Operating System – Types

iOS

Similarly to macOS, iOS is Apple’s meticulously created operating system for mobile devices. iOS has a carefully curated App Store ecosystem, a highly secure environment, and more. While iOS restricts the extent of customization available to users, it is widely known as one of the safest types of operating systems for mobile devices.

iOS operating systems can facilitate seamless integration with numerous Apple devices such as iPhones, iPads, Macs, and more. Furthermore, powerful features such as Handoff, COntinuity, and AirDrop can help users efficiently share content, information, and messages across different devices.
This can help transcend the boundaries of individual devices by creating a unified user experience.

Operating System – Types

Server Operating Systems
A server operating system refers to specialized software designed to help users manage and monitor the overall operations of a computer server. Unlike desktop or mobile operating systems, which are built for individual personal devices, server operating systems are designed specifically to manage data and resources for multiple users or clients.

Windows Server
Under the Windows Server family, various versions are specifically designed to cater to unique and diverse business requirements. These include Standard, Datacenter, and Essentials.
These different versions have powerful and robust features that can help businesses undergo centralized management, cloud integration, and virtualization tasks.
Windows Server’s dominant market presence and widespread adoption across various industries and enterprises affirm its powerful capabilities.

Linux-Based Server OS
Linux offers powerful, open-source types of operating systems that can empower businesses to manage workloads more flexibly. It includes different versions such as Ubuntu Server, CentOS, Red Hat Enterprise Linux, and more.
Linux-based server OS is popular for its highly scalable features without compromising security and encryption.

Operating System – Types

Embedded Operating Systems

Embedded Systems OS
An embedded operating system is a specialized OS designed to perform specific tasks for any device that is not a computer. Embedded Systems OS are usually used to operate everyday appliances and IoT devices such as thermostats, washing devices, etc. These types of operating systems are specifically designed to promote efficient resource allocation, enhance convenience, and promote seamless integration.

One of the most significant advantages of embedded operating systems is their ability to be tailored and customized according to the requirements of specific devices. These types of operating systems are built to accelerate the operations of appliances and devices by facilitating interaction between hardware and software.
However, while embedded systems OS may excel in their designated functionalities, their abilities are restricted to more versatile and general-purpose tasks.

Operating System – Types

Hybrid and Special-Purpose OS

Overview of Hybrid OS
Hybrid operating systems are usually a hyper-specific combination of different features from different types of operating systems. By strategically combining different elements from desktop, server, and RTOS, hybrid OS can provide users with more versatile and adaptable solutions. Therefore, a hybrid OS can support a more comprehensive range of applications.
For example, a hybrid OS may combine a desktop OS's user-friendly nature with a server OS's scalability to create a unified solution for demanding environments.

Special-Purpose OS
Special-purpose operating systems are designed to support specific tasks such as controlling medical equipment, managing network routers, and overseeing point-of-sale mechanisms. Therefore, the design of this type of operating system is usually aligned with the exact needs and requirements of its intended function, use, or device
Special-purpose OS is best suited for niche enterprise applications where exact control, governance, and dedicated performance are required for success. They can help ensure that overall operations continue without delays, excessive utilization of resources, or instability. Therefore, special-purpose OS may thrive when more generic operating systems fall short.

Operating System – Types

Real-Time OS (RTOS)
Real-Time Operating Systems (RTOS) are mainly utilized in scenarios where exact timing is paramount. This includes time-critical applications in industrial automation, aviation, and medical devices. Utilizing an RTOS in situations like these can help ensure that mission-critical tasks are performed precisely and with minimal delays.

Takeaway - Various types of operating systems software play a crucial role in shaping user interactions with their devices and servers. Integrating multiple essential features, emphasis on security and encryption, diverse versions, and community collaboration of these systems are pivotal for the seamless functioning of other devices.
Each of the four types of operating systems consists of distinct and unique features and applications that can cater to varied user requirements. When selecting the right operating system for you, it is essential to consider which one can drive productivity over time. Key factors to consider include

- **Usage Scenarios**
- **Performance and Stability**
- **Future Proofing**

As digital technology evolves exponentially, various types of OS will become increasingly flexible and adaptable. This will pave the way for enhanced, accelerated efficiency, device compatibility, and improved user experiences.

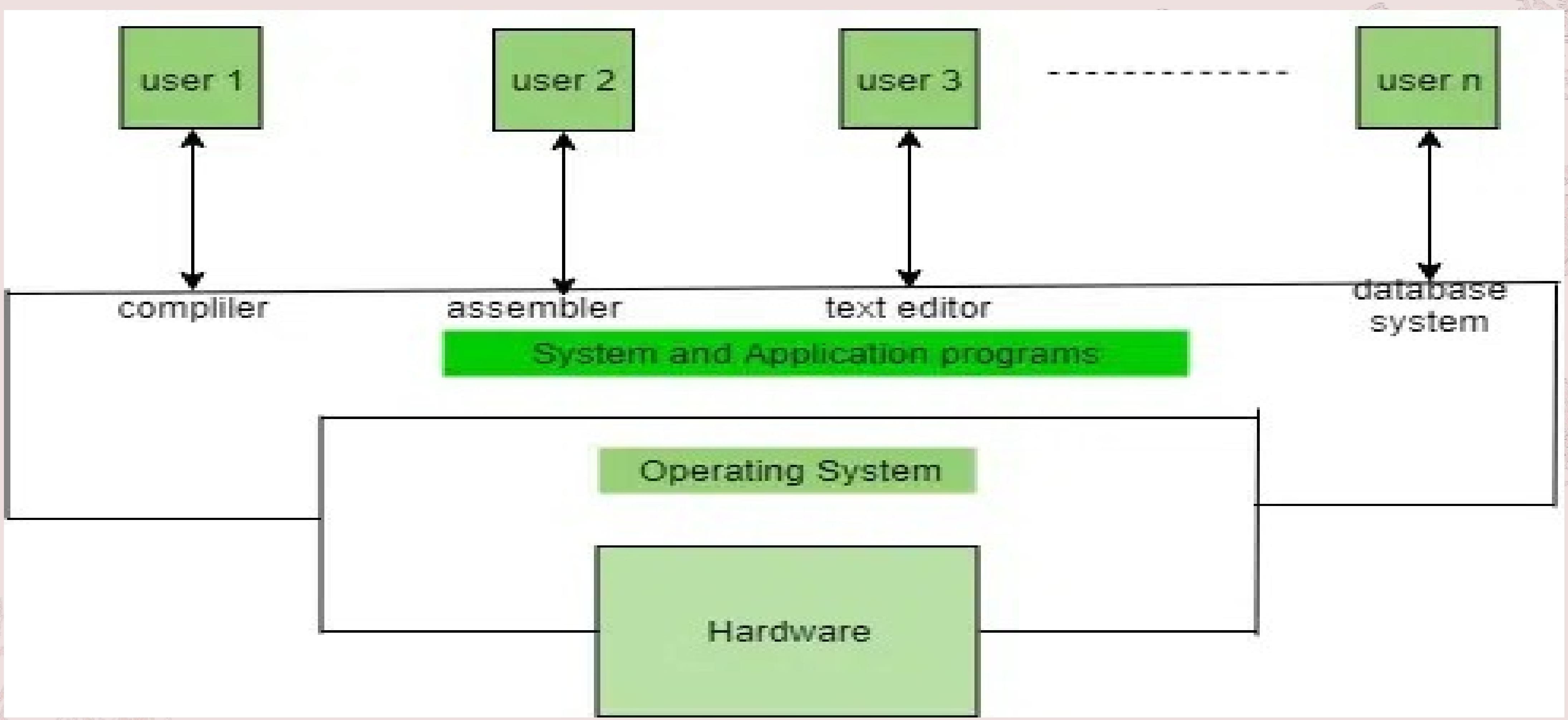
Agenda

Kernel Space	User Space
Kernels and OS core execute here.	Normal program and applications software run here.
It’s the core space of OS.	It's a form of sand-boxing that restricts user processes to access OS kernel.
It has full access to all memory and machine hardware.	It has limited access to memory and access kernel through system calls only.
It contains the page table for process, kernel data structure, threads, and kernel code etc.	It contains the program code, data, stacks, and heap of the process.

Agenda

Kernel Mode	User Mode
In kernel mode, the program has direct and unrestricted access to system resources.	In user mode, the application program do not have direct access to system resources. In order to access the resources, a system call must be made.
In Kernel mode, the whole operating system might go down if an interrupt occurs	In user mode, a single process fails if an interrupt occurs.
Kernel mode is also known as the master mode, privileged mode, or system mode.	User mode is also known as the unprivileged mode, restricted mode, or slave mode.
In kernel mode, all processes share a single virtual address space. The mode bit of kernel-mode is 0.	In user mode, all processes get separate virtual address space. While; the mode bit of user-mode is 1.
In kernel mode, the applications have more privileges as compared to user mode.	While in user mode the applications have fewer privileges.
Only essential functionality is permitted to operate in this mode.	User programs can access and execute in this mode for a given system.
The kernel mode can refer to any memory block in the system and can also direct the CPU for the execution of an instruction, making it a very potent and significant mode.	The user mode is a standard and typical viewing mode, which implies that information cannot be executed on its own or reference any memory block; it needs an Application Protocol Interface (API) to achieve these things.

Agenda


user 1
user 2
user 3
user n
compliler
assembler
text editor
database system
System and Application programs
Operating System
Hardware

Linux

What is Linux
LINUX stands for **Lovable Intellect Not Using XP**
Linux is a powerful and flexible family of operating systems that are free to use and share. It was created by a person named Linus Torvalds in 1991.

<u>Linux distribution</u> is an operating system that is made up of a collection of software based on the Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software. Around **600 + Linux Distributions** are available and some of the popular Linux distributions are:

- MX Linux
- Manjaro
- Linux Mint
- elementary
- Ubuntu
- Debian
- Solus
- Fedora
- openSUSE
- Deepin

Linux

Why Linux?

- Open source
- Access to code
- Highly secure
- Run faster
- flexible,
- Reliable
- Portable

Advantages of Linux

- Linux is compatible with a large number of file formats.
- Linux is free of cost so it is easily available for users to download and use.
- It is enhancing day by day as users can make changes in the Linux operating system.
- Linux system rarely crashes as they are very stable.
- Linux does not collect much user data, so it ensures the privacy of users.
- No reboot is needed thus increasing system speed.
- No Anti-Virus software is needed to be installed in Linux Operating System

Disadvantages of Linux

- Linux does not provide some hardware drivers which is a drawback of Linux.
- The command line interface of Linux is difficult to use for beginners.
- Some graphic tools are not available for the Linux operating system.
- Linux does not have standard versions, which makes it difficult for users to choose the best version for their needs.

Linux

What is Kernel?
The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system.

The kernel controls all essential computer operations, and provides the restriction to hardware access, coordinates all executing utilities, and manages Resources between processes. Using kernel only the user can access utilities provided by the operating system.

It manages the following resources of the Linux system –

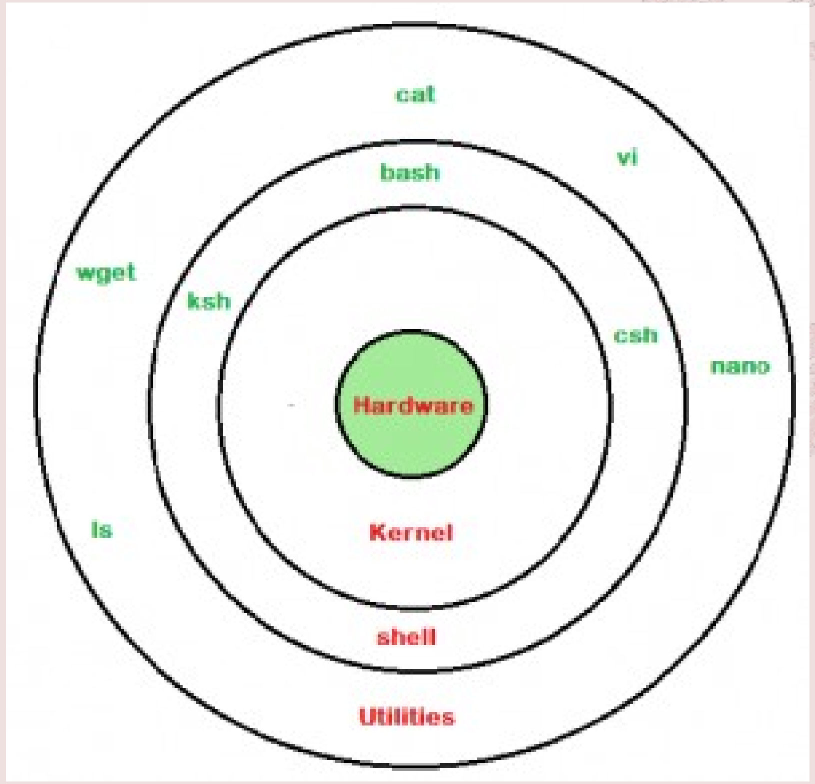
- File management
- Process management
- I/O management
- Memory management
- Device management etc.

It is often mistaken that Linus Torvalds has developed Linux OS, but actually, he is only responsible for the development of the Linux kernel.
Complete Linux system = Kernel + GNU system utilities and libraries + other management scripts + installation scripts

Linux

What is Shell?
A shell is a special user program that provides an interface for the user to use operating system services. Shell accepts human-readable commands from users and converts them into something that the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

Shell is broadly classified into two categories –
•Command Line Shell
•Graphical shell



Linux

There are several shells are available for Linux systems like –

- BASH (Bourne Again SHell) – It is the most widely used shell in Linux systems. It is used as the default login shell in Linux systems and in macOS. It can also be installed on Windows OS.

- CSH (C SHell) – The C shell's syntax and its usage are very similar to the C programming language.

- KSH (Korn SHell) – The Korn Shell was also the base for the POSIX Shell standard specifications etc.

Each shell does the same job but understands different commands and provides different built-in functions.

What is a terminal?
A program which is responsible for providing an interface to a user so that he/she can access the shell. It basically allows users to enter commands and see the output of those commands in a text-based interface. Large scripts that are written to automate and perform complex tasks are executed in the terminal.

Linux

A shell script comprises the following elements –
- Shell Keywords – if, else, break etc.
- Shell commands – cd, ls, echo, pwd, touch etc.
- Functions
- Control flow – if..then..else, case and shell loops etc.

Why do we need shell scripts?
There are many reasons to write shell scripts:
- To avoid repetitive work and automation
- System admins use shell scripting for routine backups.
- System monitoring
- Adding new functionality to the shell etc.

If you want to run a bunch of commands together, you can do so by creating a shell script. Shell scripts are very useful if you need to do a task routinely, like taking a backup. You can list those commands and execute them all with just a single script.

Shell scripts end with the extension “.sh”.
```
touch script.sh
```

Linux

Some Advantages of shell scripts

•The command and syntax are exactly the same as those directly entered in the command line, so programmers do not need to switch to entirely different syntax

•Writing shell scripts are much quicker

•Quick start

•Interactive debugging etc.

Some Disadvantages of shell scripts

•Prone to costly errors, a single mistake can change the command which might be harmful.

•Slow execution speed

•Design flaws within the language syntax or implementation

•Not well suited for large and complex task

•Provide minimal data structure unlike other scripting languages. etc.

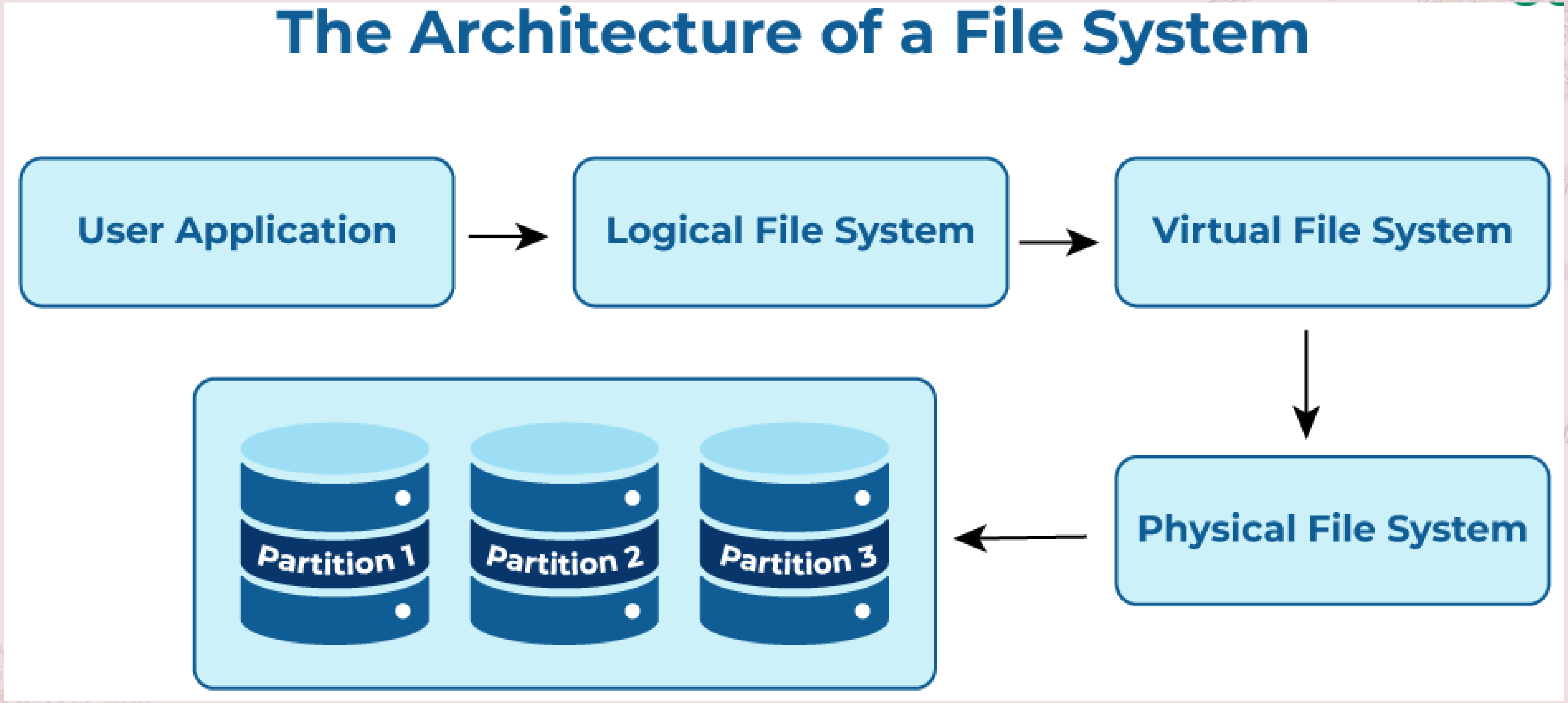
Linux

What is a File System?
A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device. Some common types of file systems include:

1. **FAT (File Allocation Table):** An older file system used by older versions of Windows and other operating systems.
2. **NTFS (New Technology File System):** A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.
3. **ext (Extended File System):** A file system commonly used on Linux and Unix-based operating systems.
4. **HFS (Hierarchical File System):** A file system used by macOS.
5. **APFS (Apple File System):** A new file system introduced by Apple for their Macs and iOS devices.

What is the Linux File System
The Linux file system is a multifaceted structure comprised of three essential layers.

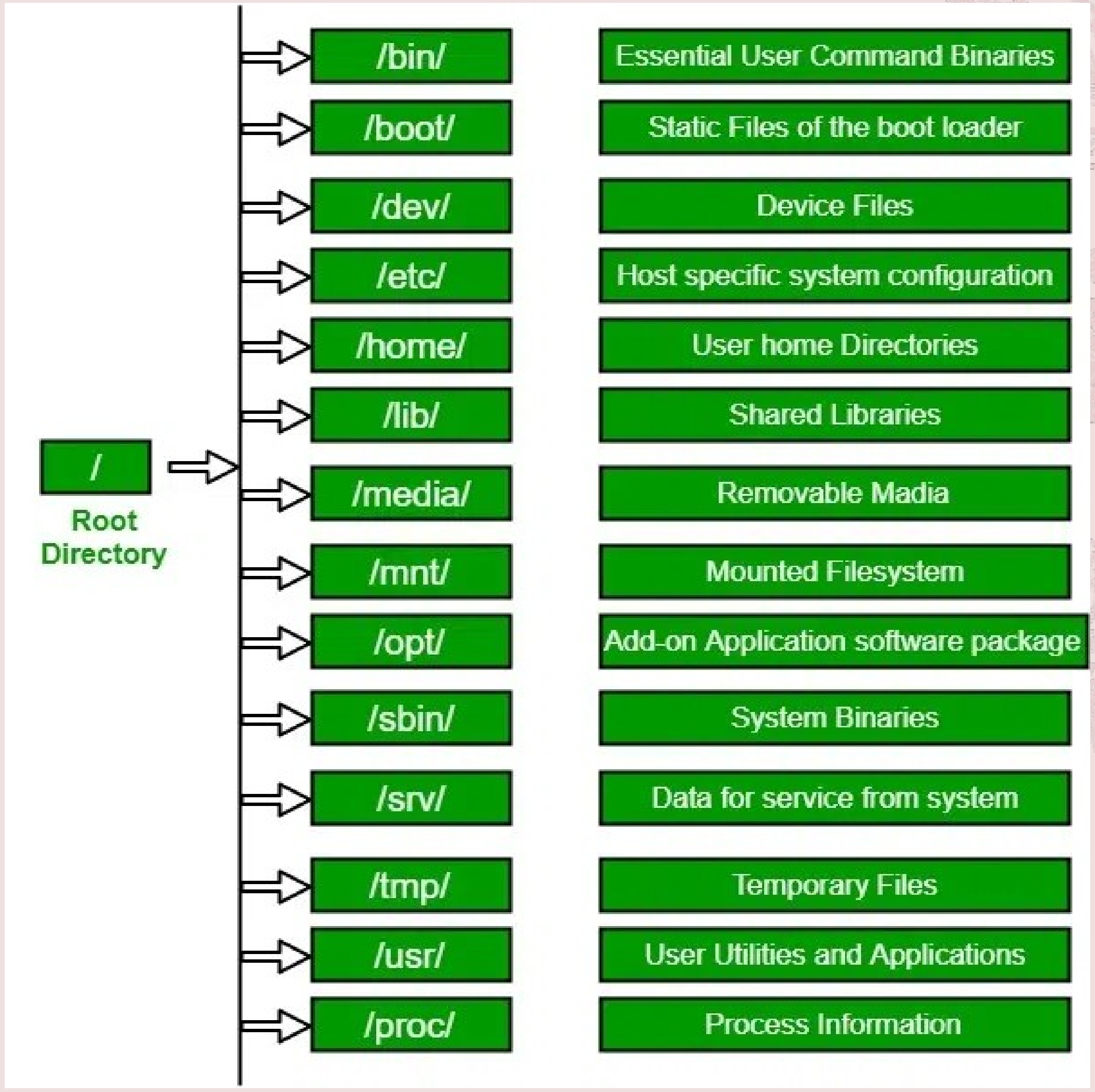
Linux


The Architecture of a File System
User Application
Logical File System
Virtual File System
Physical File System
Partition 1
Partition 2
Partition 3

Linux

Linux File Hierarchy Structure
The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.



Linux

Basic file system operations

- **File creation:** This is the act of making a new file in the file system.
- **File reading:** In this operation, the file system provides access to the contents of a file.
- **File writing:** Here, existing content in a file can be updated or new content can be added.
- **File deletion:** Deleting a file removes it from the file system.
- **Directory traversal:** This is the ability to navigate through directories or folders in the file system.

Advanced file system operations

- **File copying:** This is a process that involves creating a replica of an existing file.
- **Tagging:** This allows users to create and assign relevant labels or tags to files.
- **Locking:** This operation facilitates exclusive access rights to a file or directory.
- **Linking:** A way of making a file or directory accessible from various locations within the file system.
- **Mounting and Unmounting:** The process of adding and removing file systems from the system's file hierarchy.
- **File compression:** The act of reducing the size of files for improved storage efficiency.

Linux

A **Distributed File System (DFS)** allows users on multiple machines to share files and storage resources as if these were located on their local machine. This system makes it easier for users to access and manage files that are physically stored on other systems within the network.
The primary purpose of a DFS is to enable fast, efficient, and secure file sharing among users across a network. A DFS achieves this goal by using a client-server model where servers provide file access services, and clients consume these services. The distributed nature of the file system allows for increased availability, fault tolerance, and

•**Increased data availability:** Because files are distributed across multiple servers if one server fails, the data is still accessible from others.

Distributed File System (DFS) allows users on multiple machines to share files and storage resources as if they were located on their local machine. This system makes it easier for users to access and manage files that are physically stored on other systems within the network.
The primary purpose of a DFS is to enable fast, efficient, and secure file sharing among users across a network. A DFS achieves this goal by using a client-server model where servers provide file access services, and clients consume these services. The distributed nature of the file system allows for increased availability, fault tolerance

Linux

Advantages:

- **proved performance:** File requests can be processed by multiple servers concurrently, resulting in faster response times for users.
- **Scalability:** More servers can be added to share the load as demand increases.
- **Cost-effectiveness:** With a DFS, there is no need for high-capacity storage on each user’s device, which can provide significant cost savings.

Disadvantages:

- **Complexity:** Ensuring data consistency across all servers in the DFS can be complex and challenging to achieve in real-time.
- **Dependency on the network:** If the network experiences latency or goes down, this can affect the speed and availability of data access.
- **Security:** As data is distributed across various servers, ensuring proper security measures and controls can be demanding.

OS-Memory Management

What is Computer Memory?

The essential component of the computer is its Memory.
It is assembled on the motherboard as it is a storage device used for storing data and instructions for performing a task on the system.

Computer memory can be defined as a collection of some data represented in binary format

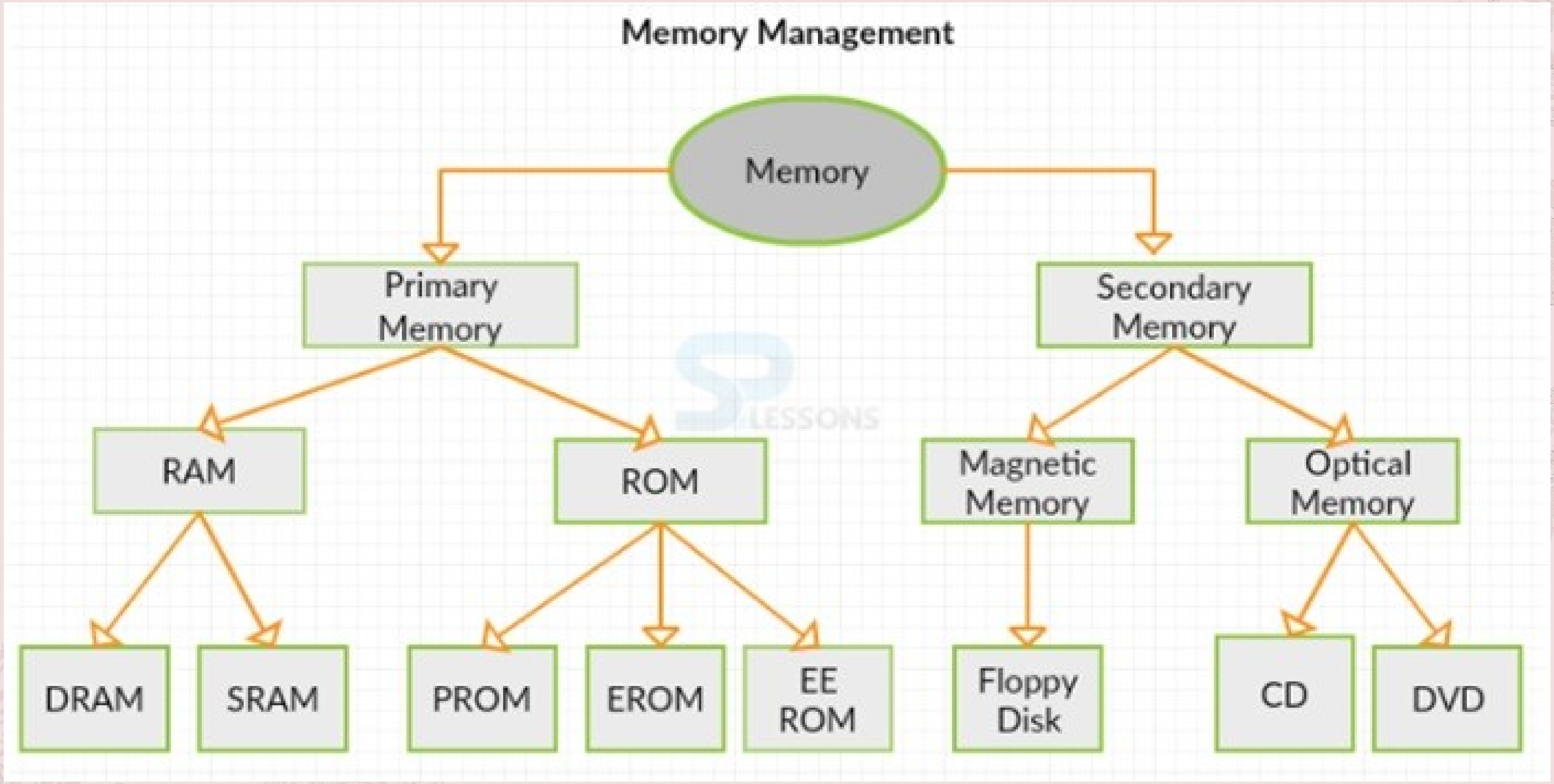
A memory is used to store data and instructions. It is the storage space in a computer, where data is to be processed and all the instructions are stored that are required for processing. In simple words, it works just like a human brain where we can store memories.

Without memory, the computer can't perform a single task.

Types of Computer Memory

1. Primary Memory
2. Secondary Memory

OS-Memory Management



OS-Memory Management

Primary Memory (Main Memory)

It is also referred to as Main Memory. It is volatile. The reason behind this is that primary memory holds only those data and instructions on which the computer is currently working that is it does not store the data permanently.

- It also stores the operating system and data required to run the computer.
- It is a limited capacity memory and data, or information is lost when power is switched off. Primary Memory is generally constructed with a semiconductor device.
- Registers are much faster than these memories, but it is faster than secondary memory.
- It contains all the data and instructions that are required to be processed.

It is further divided into two subcategories **RAM and ROM**.

OS-Memory Management

RAM (Random Access Memory)

- It is Random Access Memory because of the random selection of memory locations.
- It performs both read and write operations on memory. It stores data temporarily.
- If power failures happen in the system during memory access, then you will lose your data permanently. Hence RAM is a volatile memory.

SRAM (Static Random Access Memory)

- It holds data in a static form.
- Static RAM provides faster access to data and is more expensive than DRAM as each cell must contain multiple transistors. SRAM does not use capacitors.
- It is highly recommended for use in PCs, peripheral equipment, printers, LCD screens, and hard disks.

DRAM (Dynamic Random Access Memory)

- This type of memory uses separate capacitors or transistors to store each bit of data and it has two states of value in one bit called 0 and 1.
- As compared with other RAM it is less expensive. DRAM requires less power than other RAMs.
- In DRAM, data is written at the byte level and it reads data at the multiple-byte page level.

OS-Memory Management

ROM (Read Only Memory)

- ROM offers huge types of standards to save data as it is a permanent memory location.
- It works with the read-only operation. whenever power failure occurs during the ROM memory work in computers then no data loss happens.
- It is used where the programming requires no change and also in embedded systems
- It is used in peripheral devices and calculators.

PROM (Programmable read-only memory)

- It can be coded by the user. Once coded, the data and instructions in it cannot be changed.
- Creating ROM chips from scratch is time-consuming and very expensive.
- It is used to store permanent data in digital electronic devices.
- It can be bought at a low cost as compared to other RAMs.

EPROM (Erasable Programmable read-only memory)

- This is the type of memory that can be reprogrammed.
- We can erase data from it and reprogram it that is erase all the previous data by using high-voltage Ultraviolet light. It is required to erase each cell in EPROM.

OS-Memory Management

EEPROM (Electrically erasable programmable read-only memory)

The data can be erased and reprogrammed by applying an electric charge.
There is no need for ultraviolet light, and we can erase only portions of the chip.
It was a replacement for PROM and EPROM chips and later it was used for the computer's BIOS.
Configuration parameters are stored by using EEPROM.
It is required that data be written or erased by EEPROM one byte at a time.

Secondary Memory

The secondary memory is non-volatile, persistent, and not immediately accessible by a computer.
By using secondary memory, we can store, retrieve, and transmit the data and information.

Magnetic Memory: Examples of magnetic memory are floppy diskettes, hard disks, tape drives, etc.

Optical Memory: Examples of optical memory are CD-ROMs, DVDs, Virtual optical devices, RDX, Flash drives, etc.

OS-Memory Management

Why is Memory Management Required?

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

Contiguous and Noncontiguous Memory Allocation

Memory is a huge collection of bytes, and memory allocation refers to allocating space to computer applications.
Mainly two types of memory allocation: contiguous and non-contiguous memory allocation.

Contiguous Memory Allocation: Contiguous memory allocation is a method in which a single contiguous section/part/space of memory is allocated to a process or file needing it.

Advantages
1.It is simple to keep track of how many memory blocks are left, which determines how many more processes can be granted memory space.
2.The read performance of contiguous memory allocation is good because the complete file may be read from the disk in a single task.
3.The contiguous allocation is simple to set up and performs well.

Disadvantages
4.Fragmentation isn't a problem because every new file may be written to the end of the disk after the previous one.
5.When generating a new file, it must know its eventual size to select the appropriate hole size.
6.When the disk is filled up, it would be necessary to compress or reuse the spare space in the holes.

Contiguous and Noncontiguous Memory Allocation

Noncontiguous memory allocation assigns the method to distinct memory sections at numerous memory locations.

Advantages

1.It has the advantage of reducing memory waste, but it increases overhead because of the address translation.

2.It slows down the memory execution because time is consumed in address translation.

Disadvantages

3.The downside of this memory allocation is that the access is slow because you must reach the other nodes using pointers and traverse them.

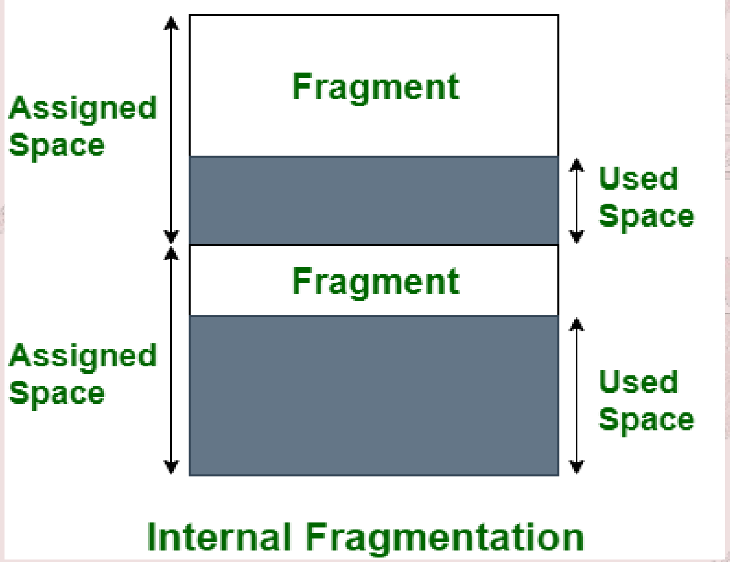
Contiguous and Noncontiguous Memory Allocation

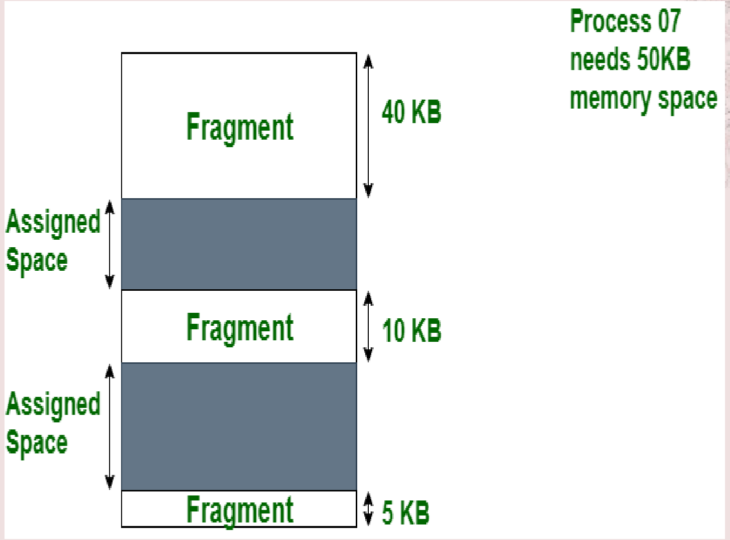
Contiguous Memory Allocation	Non-Contiguous Memory Allocation
Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
Faster in Execution. and t is easier for the OS to control.	Slower in Execution and It is difficult for the OS to control.
Swapped-in processes are arranged in the originally allocated space.	Swapped-in processes can be arranged in any place in the memory.
Overhead is minimal as not many address translations are there while executing a process.	More Overheads are there as there are more address translations.
Both Internal fragmentation and external fragmentation occurs in Contiguous memory allocation method.	Only External fragmentation occurs in Non-Contiguous memory allocation method.
It includes single-partition and multi-partition allocation. Fixed(or static) partitioning Dynamic partitioning	It includes paging and segmentation. Multilevel Paging /Inverted Paging Segmentation / Segmented Paging
It could be visualized and implemented using Arrays and Degree of multiprogramming is fixed as fixed partitions	It could be implemented using Linked Lists. Degree of multiprogramming is not fixed

OS-Memory Management

Internal Fragmentation
Internal fragmentation happens when the memory is split into mounted-sized blocks. Whenever a method is requested for the memory, the mounted-sized block is allotted to the method. In the case where the memory allotted to the method is somewhat larger than the memory requested, then the difference between allotted and requested memory is called internal fragmentation. We fixed the sizes of the memory blocks, which has caused this issue. If we use dynamic partitioning to allot space to the process, this issue can be solved.

External Fragmentation
External fragmentation happens when there's a sufficient quantity of area within the memory to satisfy the memory request of a method. However, the process's memory request cannot be fulfilled because the memory offered is in a non-contiguous manner. Whether you apply a first-fit or best-fit memory allocation strategy it'll cause external fragmentation.
Here, we use compaction, paging, or segmentation to use the free space to run a process.





OS-Memory Management

Internal fragmentation	External fragmentation
In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to the method.
Internal fragmentation happens when the method or process is smaller than the memory.	External fragmentation happens when the method or process is removed.
The solution of internal fragmentation is the best-fit block.	The solution to external fragmentation is compaction and paging.
Internal fragmentation occurs when memory is divided into fixed-sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, which is called External fragmentation
Internal fragmentation occurs with paging and fixed partitioning.	External fragmentation occurs with segmentation and dynamic partitioning.
It occurs on the allocation of a process to a partition greater than the process's requirement. The leftover space causes degradation system performance.	It occurs on the allocation of a process to a partition greater which is exactly the same memory space as it is required.
It occurs in worst fit memory allocation method.	It occurs in best fit and first fit memory allocation method

OS-Memory Management

Compaction

Compaction is a technique to collect all the free memory present in the form of fragments into one large chunk of free memory, which can be used to run other processes.
It does that by moving all the processes towards one end of the memory and all the available free space towards the other end of the memory so that it becomes contiguous.
It is not always easy to do compaction. Compaction can be done only when the relocation is dynamic and done at execution time. Compaction can not be done when relocation is static and is performed at load time or assembly time.

OS-Memory Management

Purpose of Compaction in Operating System

While allocating memory to a process, the operating system often faces a problem when there’s a sufficient amount of free space within the memory to satisfy the memory demand of a process. However, the process’s memory request can’t be fulfilled because the free memory available is in a non-contiguous manner, this problem is referred to as external fragmentation. To solve such kinds of problems compaction technique is used.

Issues with Compaction

Although the compaction technique is very useful in making memory utilization efficient and reduces external fragmentation of memory, the problem with it is that a large amount of time is wasted in the process and during that time the CPU sits idle hence reducing the efficiency of the system.

OS-Memory Management

Advantages of Compaction

- Reduces external fragmentation.
- Make memory usage efficient.
- Memory becomes contiguous.
- Since memory becomes contiguous more processes can be loaded to memory, thereby increasing the scalability of OS.
- Fragmentation of the file system can be temporarily removed by compaction.
- Improves memory utilization as there is less gap between memory blocks.

Disadvantages of Compaction

- System efficiency is reduced, and latency is increased.
- A huge amount of time is wasted in performing compaction.
- CPU sits idle for a long time.
- Not always easy to perform compaction.
- It may cause deadlocks since it disturbs the memory allocation process.

Agenda

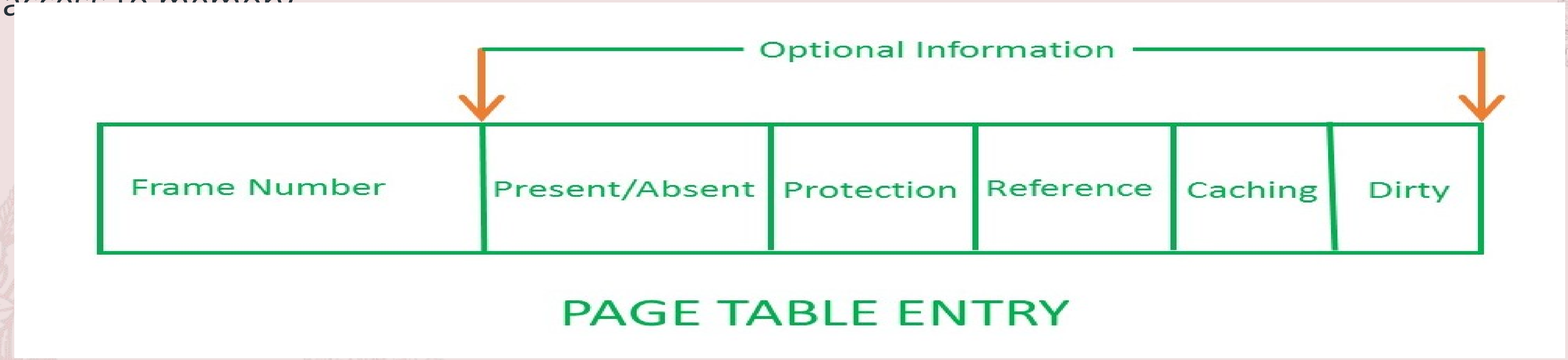
Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's logical pages to the physical page frames.

The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.

OS-Memory Management

A **Page Table** is a data structure used by the operating system to keep track of the mapping between virtual addresses used by a process and the corresponding physical addresses in the system's memory.
A Page Table Entry (PTE) is an entry in the Page Table that stores information about a particular page of memory. Each PTE contains information such as the physical address of the page in memory, whether the page is present in memory or not, whether it is writable or not, and other access permissions.
The size and format of a PTE can vary depending on the architecture of the system and the operating system used. In general, a PTE contains enough information to allow the operating system to manage memory efficiently and protect the system from malicious or accidental access to memory.



OS-Memory Management

What is an Inverted Page Table?

In an operating system that uses virtual memory, an Inverted Page Table (IPT) is a data structure used to map physical memory pages to virtual memory pages. Unlike a traditional Page Table, which is a per-process data structure, an IPT is a system-wide data structure that contains an entry for each physical page in memory.

An alternate approach is to use the **Inverted Page Table** structure that consists of a one-page table entry for every frame of the main memory. So the number of page table entries in the Inverted Page Table reduces to the number of frames in physical memory and a single page table is used to represent the paging information of all the processes.

OS-Memory Management

Segmentation in Operating System

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

Types of Segmentation in Operating Systems

- **Virtual Memory Segmentation:** Each process is divided into several segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.
- **Simple Segmentation:** Each process is divided into several segments, all of which are loaded into memory at run time, though not necessarily contiguously

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called a Segment Table.

OS-Memory Management

What is a Segment Table?

It maps a two-dimensional Logical address into a one-dimensional Physical address.

It’s each table entry has:

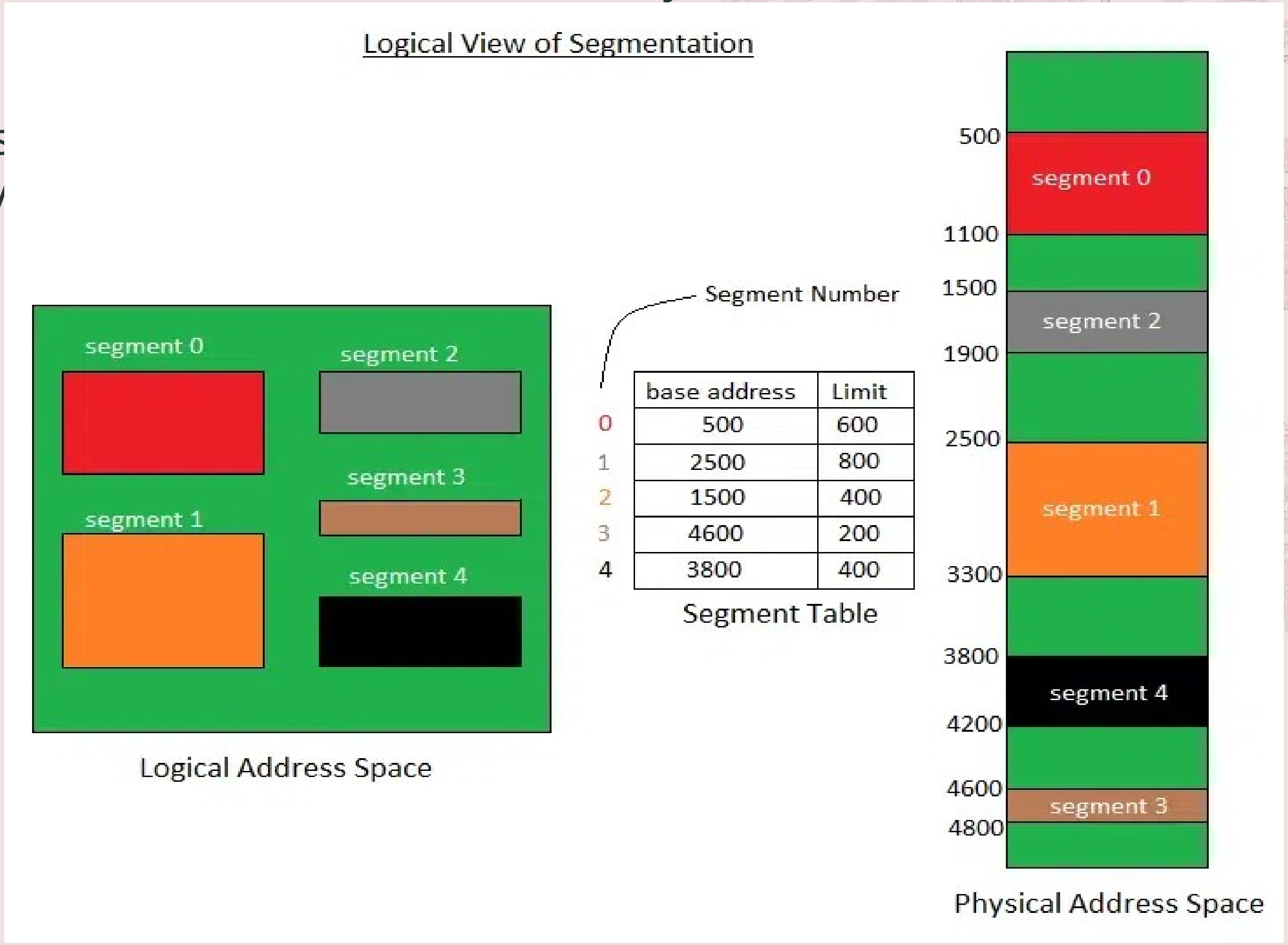
•**Base Address:**

It contains the starting physical addres
where the segments reside in memory

•**Segment Limit:**

•Also known as segment offset.

It specifies the length of the segment.



	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

OS-Memory Management

What is a dirty bit?
A dirty bit, also known as a modified bit or write bit, is a flag that is used in computer systems to indicate whether a particular memory address or disk block has been modified since it was last written to. It is an important concept in computer science and plays a crucial role in various areas such as caching, virtual memory management, and file system operations.

How does the dirty bit work?
When a process modifies a memory address or writes data to a disk block, the dirty bit for that address or block indicates it has been changed. This allows the system to keep track of which portions of memory or disk need to be saved or written back to secondary storage when resources become scarce or when a shutdown occurs. The dirty bit is typically stored alongside other bits in a control or metadata structure associated with the memory address or disk block.

What Is Reentrant Code?
Reentrant (multi-instance) code is a reusable routine that multiple programs can invoke, interrupt, and reinvoke simultaneously. When you want to reuse code, but associate each instance of the shared code with unique, preserved data, use reentrant code.

OS-Memory Management

CPU Throttling

Adjusting the clock speed of the CPU. Also called "dynamic frequency scaling," CPU throttling is commonly used to automatically slow down the computer when possible to use less energy and conserve battery, especially in laptops. CPU throttling can also be adjusted manually to make the system quieter because the fan can then run slower.

Agenda

Partitioning Algorithms
Various algorithms are implemented by the Operating System to find out the holes in the linked list and allocate them to the processes.

First Fit Algorithm
First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and loads the process into that hole.

Next Fit Algorithm
The next Fit algorithm is similar to the First Fit algorithm except for the fact that the next fit scans the linked list from the node where it previously allocated a hole.

Best Fit Algorithm
The Best Fit algorithm tries to find the smallest hole possible in the list that can accommodate the size requirement of the process.

Worst Fit Algorithm
The worst fit algorithm scans the entire list every time and tries to find out the biggest hole in the list which can fulfill the requirement of the process.

Process

SSH
SSH which stands for Secure Shell, It is used to connect to a remote computer securely. Compare to Telnet, SSH is secure wherein the client /server connection is authenticated using a digital certificate and passwords are encrypted. Hence it’s widely used by system administrators to control remote Linux servers.

Syntax: SSH username@ip-address or hostname

Ping
This utility is commonly used to check whether your **connection to the server** is healthy or not.This command is also used in –
•Analyzing network and host connections
•Tracking network performance and managing it
•Testing hardware and software issues

Syntax: ping hostname=“” or=“”
Example : ping 172.16.170.1

Agenda

FTP

FTP **is file transfer protocol**. It’s the **most preferred protocol for data transfer** amongst computers.

You can use FTP to –

•Logging in and establishing a connection with a remote host

•Upload and download files

•Navigating through directories

•Browsing contents of the directories

Telnet

There are times when we are required to connect to a remote Unix machine and work on that machine remotely. **Telnet** is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you login using Telnet, you can perform all the activities on your remotely connected machine.

finger

The **finger** command displays information about users on a given host. The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

ACL

Access Control Lists(ACL) in Linux
What is ACL ?
Access control list (ACL) provides an additional, more flexible permission mechanism for file systems. It is designed to assist with UNIX file permissions. ACL allows you to give permissions for any user or group to any disc resource.
Use of ACL :
Think of a scenario in which a particular user is not a member of group created by you but still you want to give some read or write access, how can you do it without making user a member of group, here comes in picture Access Control Lists, ACL helps us to do this trick.
Basically, ACLs are used to make a flexible permission mechanism in Linux.
From Linux man pages, ACLs are used to define more fine-grained discretionary access rights for files and directories.

setfacl and **getfacl** are used for setting up ACL and showing ACL respectively.
getfacl test/declarations.h

Shell Scripting

1) To add permission for user

```
setfacl -m "u:user:permissions" /path/to/file
```

2) To add permissions for a group

```
setfacl -m "g:group:permissions" /path/to/file
```

3) To allow all files or directories to inherit ACL entries from the directory it is within

```
setfacl -dm "entry" /path/to/dir
```

4) To remove a specific entry

```
setfacl -x "entry" /path/to/file
```

Shell Scripting

Shell Scripting – Shell Variables
A shell variable is a character string in a shell that stores some value. It could be an integer, filename, string, or some shell command itself.

Rules for variable definition
A variable name could contain any alphabet (a-z, A-Z), any digits (0-9), and an underscore (_). However, a variable name must start with an alphabet or underscore. It can never start with a number. There should be no whitespace on either side of the assignment operator(=)
Variable names cannot be reserved words. Variable names cannot have whitespace in between

Valid Variable Names
ABC
_AV_3
AV232

Invalid variable names
2_AN
!ABD
$ABC
&QAID

Agenda

Local Variable:
Variables which are specific to the current instance of shell. They are basically used within the shell, but not available for the program or other shells that are started from within the current shell.

`name=Jayesh`

Environment Variable:
These variables are commonly used to configure the behavior script and programs that are run by shell. Environment variables are only created once, after which they can be used by any user.

`export PATH=/usr/local/bin:$PATH` would add `/usr/local/bin` to the beginning of the shell's search path for executable programs

Shell Variables:
Variables that are set by shell itself and help shell to work with functions correctly. It contains both, which means it has both, some variables are Environment variable, and some are Local Variables

`$PWD` = Stores working directory
`$HOME` = Stores user's home directory
`$SHELL` = Stores the path to the shell program that is being used.

Agenda

Metacharacters: These are the special characters that are first interpreted by the shell before passing the same to the command. They are also known as shell wildcards.

- **$** Variable Substitution or expand the value of Variable.
- **>** used for Output Redirection.
- **>>** used for Output Redirection to append.
- **<** Input redirection.
- **<<** used for input redirection and is also known as here document.
- ***** Match any number of characters, Substitution wildcard for zero or more characters
- **?** Match one character, Substitution wildcard for 1 character
- **[]** Match range of characters, Substitution wildcard for any character between brackets
- **\`cmd\`** Replace cmd with the command to execute and will execute that, Substitution wildcard for command execution
- **$(cmd)** Replace cmd with the command to execute and will execute that, Substitution wildcard for command execution
- **|** Pipe is a Redirection to send the output of one command/program/process to another command/program/process for further processing.
- **;** Command separator is used to execute 2 or more commands with one statement.
- **||** OR conditional execution of the commands.
- **&&** AND conditional execution of the commands.
- **()** Groups the command into one output stream.
- **&** executes command in the background and will display the assigned Pid.
- **#** to comment something.
- \ used to escape the interpretation of a character or to prevent that.

Agenda

The wildcard characters

Wildcard characters provide a convenient way to specify multiple file names or directory names.

The wildcard characters are asterisk (*) and question mark (?). The metacharacters are open and close square brackets ([ ]), hyphen (-), and exclamation mark (!).

Pattern matching using the * wildcard character
Use the asterisk (*) to match any sequence or string of characters.
Pattern matching using the ? wildcard character
Use the ? to match any one character.
Pattern matching using [ ] shell metacharacters
Metacharacters offer another type of wildcard notation by enclosing the desired characters within [ ]. It is like using the ?, but it allows you to choose specific characters to be matched.

Agenda

Following are the types of loop statements that are available in Shell Scripting:

- While loop,
- For loop, and.
- Until loop.

- The **if...else** statement

- if...fi statement
- if...else...fi statement
- if...elif...else...fi statement

- The **case...esac** statement

Agenda

Operator	Description
==	Returns true if the strings are equal
!=	Returns true if the strings are not equal
-n	Returns true if the string to be tested is not null
-z	Returns true if the string to be tested is null

Operator	Description
-eq	Equal
-ge	Greater Than or Equal
-gt	Greater Than
-le	Less Than or Equal
-lt	Less Than
-ne	Not Equal

Agenda

Environment Variables	Description
$USER	Gives search path for commands.
$PATH	Gives search path for commands.
$HOME	Gives path of home directory.
$PWD	Gives the path of present working directory.
$HOSTNAME	Gives name of the host.
$LANG	Gives the default system language.
$EDITOR	Gives default file editor.
$UID	Gives user ID of current user.
$SHELL	Gives location of current user’s shell program.

Agenda

Understanding Different Parts of BASH Prompt
To customize the bash prompt, first, we should understand how the bash prompt works. Bash provides the Prompt Statement. There are four bash prompt statement
•**PS1 –** This is the primary prompt statement. We will customize this prompt.
•**PS2 –** This is the secondary prompt statement. Basically, it is used when the user provides the long command separated by \ characters.
•**PS3 –** This prompt is used to select the command.
•**PS4 –** This prompt is used for running a shell script in debug mode.

Process

What is process
A process refers to the active execution of a program. It consists of several components, including data retrieved from files, user input, program instructions, etc.

A process is a program in execution.
For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we run the binary code, it becomes a process.
A process is an ‘active’ entity instead of a program, which is considered a ‘passive’ entity.
A single program can create many processes when run multiple times

Process management refers to the techniques and strategies used by organizations to design, monitor, and control their business processes to achieve their goals efficiently and effectively. It involves identifying the steps involved in completing a task, assessing the resources required for each step, and determining the best way to execute the task.

Process management can help organizations improve their operational efficiency, reduce costs, increase customer satisfaction, and maintain compliance with regulatory requirements. It involves analyzing the performance of existing processes, identifying bottlenecks, and making changes to optimize the process flow.

Process

Preemptive Scheduling

Preemptive scheduling is used when a process switches from the running state to the ready state or from the waiting state to the ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.

What are the different terminologies to take care of in any CPU Scheduling algorithm?

- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time:** Time at which process completes its execution.
- **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time:** Time Difference between completion time and arrival time
- **Waiting Time(W.T):** Time Difference between turn around time and burst time.

Process

Shortest Remaining Time First (SRTF) scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a **smaller amount of time.**

Preemptive Scheduling Advantages

1.Because a process may not monopolize the processor, it is a more reliable method.
2.Each occurrence prevents the completion of ongoing tasks.
3.The average response time is improved.
4.Utilizing this method in a multi-programming environment is more advantageous.
5.The operating system makes sure that every process using the CPU is using the same amount of CPU time.

Preemptive Scheduling Disadvantages

6.Limited computational resources must be used.
2.Suspending the running process, changing the context, and dispatching the new incoming process all take more time.
3.The low-priority process would have to wait if multiple high-priority processes arrived at the same time.

Process

Shortest Job First(SJF):
is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.
Characteristics of SJF:
•Shortest Job First has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
•It is associated with each task as a unit of time to complete.
•It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.

Longest Job First(LJF):
The scheduling process is just the opposite of the shortest job first (SJF), as the name suggests this algorithm is based on the fact that the process with the largest burst time is processed first. Longest Job First is non-preemptive in nature.
Characteristics of LJF:
•Among all the processes waiting in a waiting queue, the CPU is always assigned to the process having the largest burst time.
•If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first. And LJF CPU Scheduling can be of both preemptive and non-preemptive types.

Process

Priority Scheduling:
is a preemptive method of CPU scheduling algorithm that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there is more than one process with equal value, then the most important CPU planning algorithm works based on the FCFS

Characteristics of Priority Scheduling:
•Schedules tasks based on priority.
•When the higher priority work arrives and a task with less priority is executed, the higher priority process will take the place of the less priority process
•The later is suspended until the execution is complete.
• The lower the number assigned, the higher the priority level of a process.

Round robin:
is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of the First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round robin:
•It’s simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
•One of the most widely used methods in CPU scheduling as a core.
•It is considered preemptive as the processes are given to the CPU for a very limited time.

Process

Non-Preemptive Scheduling
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.

First Come First Serve:
FCFS considered to be the simplest of all operating system scheduling algorithms. The first come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using a FIFO queue.
Characteristics of FCFS:
- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not very efficient in performance, and the wait time is quite high.

Process

Advantages
1.It has a minimal scheduling burden.
2.It is a very easy procedure.
3.Less computational resources are used.
4.It has a high throughput rate.

Disadvantages
5.Its response time to the process is super.
2.Bugs can cause a computer to freeze up.

Process

Shortest Remaining Time First:

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

•SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given it's overhead charges are not counted.

•The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.

Longest Remaining Time First:

The longest remaining time first is a preemptive version of the longest job first scheduling algorithm. This scheduling algorithm is used by the operating system to program incoming processes for use in a systematic way. This algorithm schedules those processes first which have the longest processing time remaining for completion.

Characteristics of longest remaining time first:

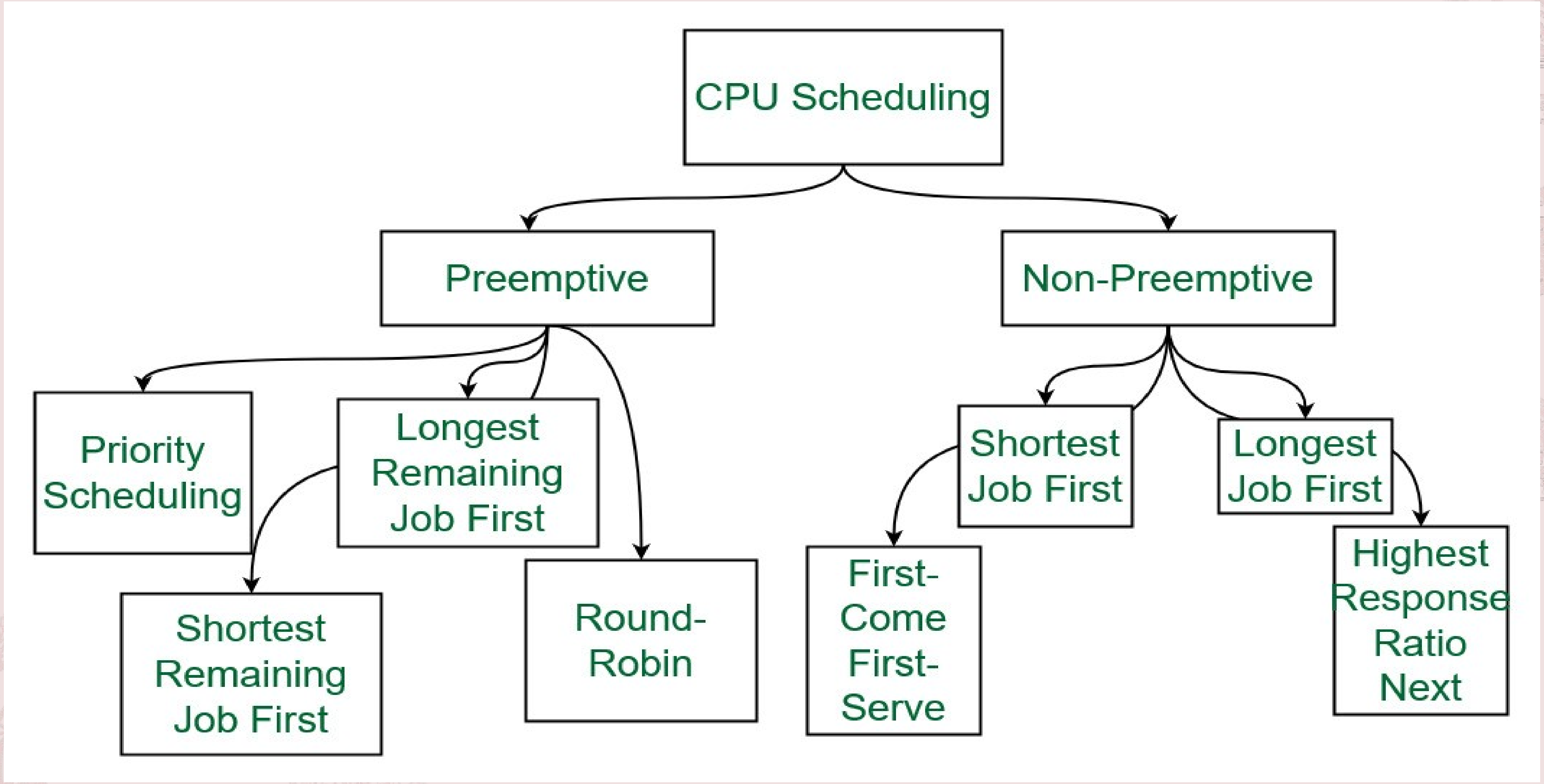
•Among all the processes waiting in a waiting queue, the CPU is always assigned to the process having the largest burst time.

•If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first. And LRTF CPU Scheduling can be of both preemptive and non-preemptive.

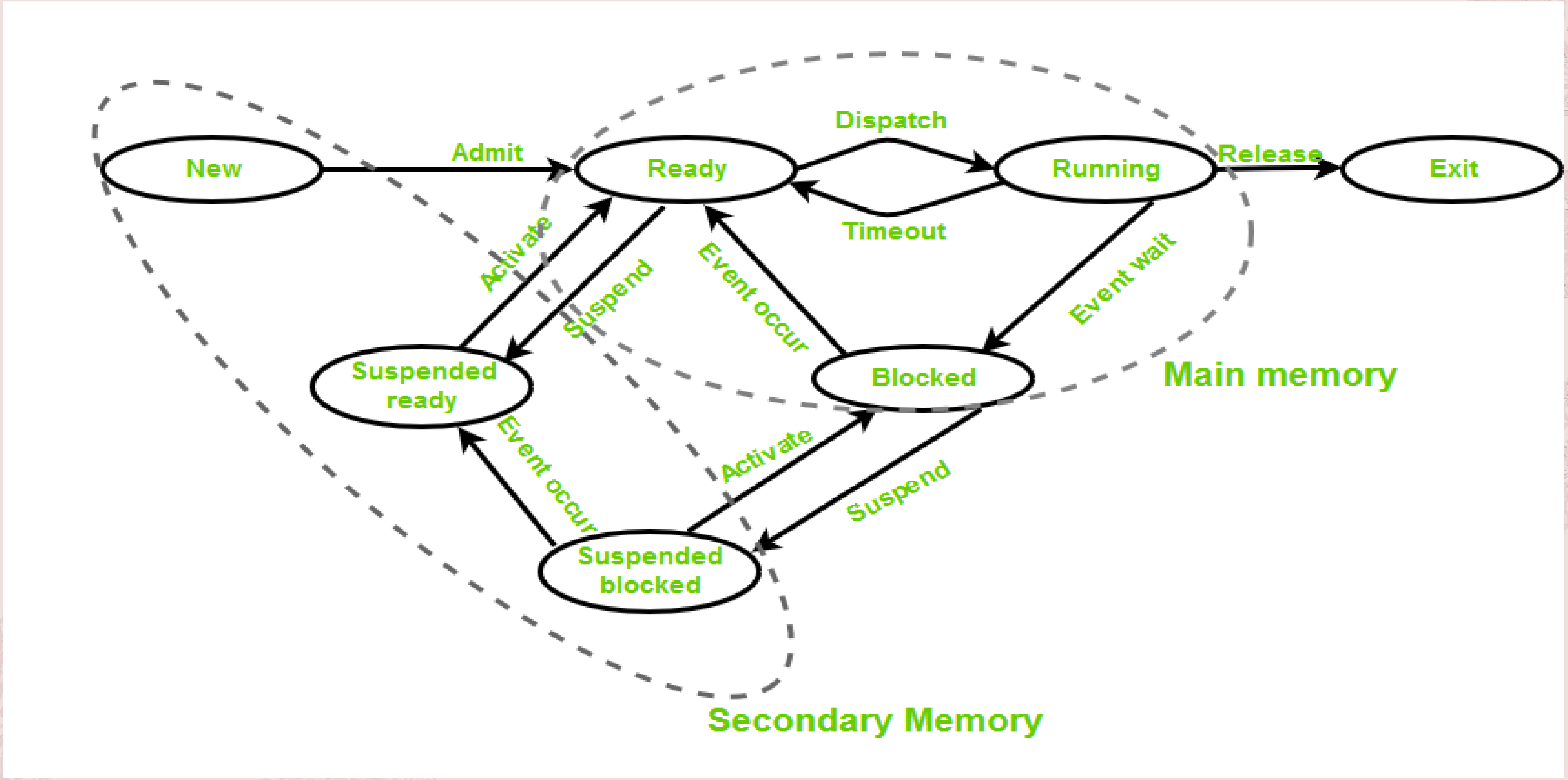
Process

Highest Response Ratio Next:
Highest Response Ratio Next is a non-preemptive CPU Scheduling algorithm and it is considered as one of the most optimal scheduling algorithms. The name itself states that we need to find the response ratio of all available processes and select the one with the highest Response Ratio. A process once selected will run till completion.
Characteristics of Highest Response Ratio Next:
•The **criteria** for HRRN is **Response Ratio,** and the **mode** is **Non-Preemptive.**
•HRRN is considered as the modification of Shortest Job First to reduce the problem of starvation.
•In comparison with SJF, during the HRRN scheduling algorithm, the CPU is allotted to the next process which has the **highest response ratio** , and not to the process having less burst time.

Process


CPU Scheduling
Preemptive
Non-Preemptive
Priority Scheduling
Longest Remaining Job First
Shortest Remaining Job First
Round-Robin
Shortest Job First
Longest Job First
First-Come First-Serve
Highest Response Ratio Next

Process states



Process

States of Process

- **New:** Newly Created Process (or) being-created process.
- **Ready:** After the creation process moves to the Ready state, i.e. the process is ready for execution.
- **Run:** Currently running process in CPU (only one process at a time can be under execution in a single processor)
- **Wait (or Block):** When a process requests I/O access.
- **Complete (or Terminated):** The process completed its execution.
- **Suspended Ready:** When the ready queue becomes full, some processes are moved to a suspended ready state
- **Suspended Block:** When the waiting queue becomes full.

Process

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: The short-term scheduler will decide which process is to be executed next and then it will call the dispatcher. A dispatcher is software that moves the process from ready to run and vice versa. In other words, it is context switching. It runs frequently. The short-term scheduler is also called the 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 the degree of multiprogramming.

Process

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 process execution, till the CPU releases the control by itself, control cannot be taken back forcibly from the CPU.

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.

Process

Fork system call is an essential operation.
The fork system call allows the creation of a new process. When a process calls the fork(), it duplicates itself, resulting in two processes running at the same time. The new process that is created is called a child process. It is a copy of the parent process. The fork system call is required for process creation and enables many important features such as parallel processing, multitasking, and the creation of complex process hierarchies.

Process

Terminologies Used in Fork System Call in Operating System

- **Process:** In an operating_system, a process is an instance of a program that is currently running. It is a separate entity with its own memory, resources, CPU, I/O hardware, and files.
- **Parent Process:** The process that uses the fork system call to start a new child process is referred to as the parent process. It acts as the parent process's beginning point and can go on running after the fork.
- **Child Process:** The newly generated process as a consequence of the fork_system call is referred to as the child process. It has its own distinct process ID (PID), and memory, and is a duplicate of the parent process.
- **Process ID:** A process ID (PID) is a special identification that the operating system assigns to each process.
- **Copy-on-Write:** The fork system call makes use of the memory management strategy known as copy-on-write. Until one of them makes changes to the shared memory, it enables the_parent_and_child_processes to share the same physical memory. To preserve data integrity, a second copy is then made.
- **Return Value:** The fork system call's return value gives both the parent and child process information. It assists_in handling mistakes during process formation and determining the execution route.

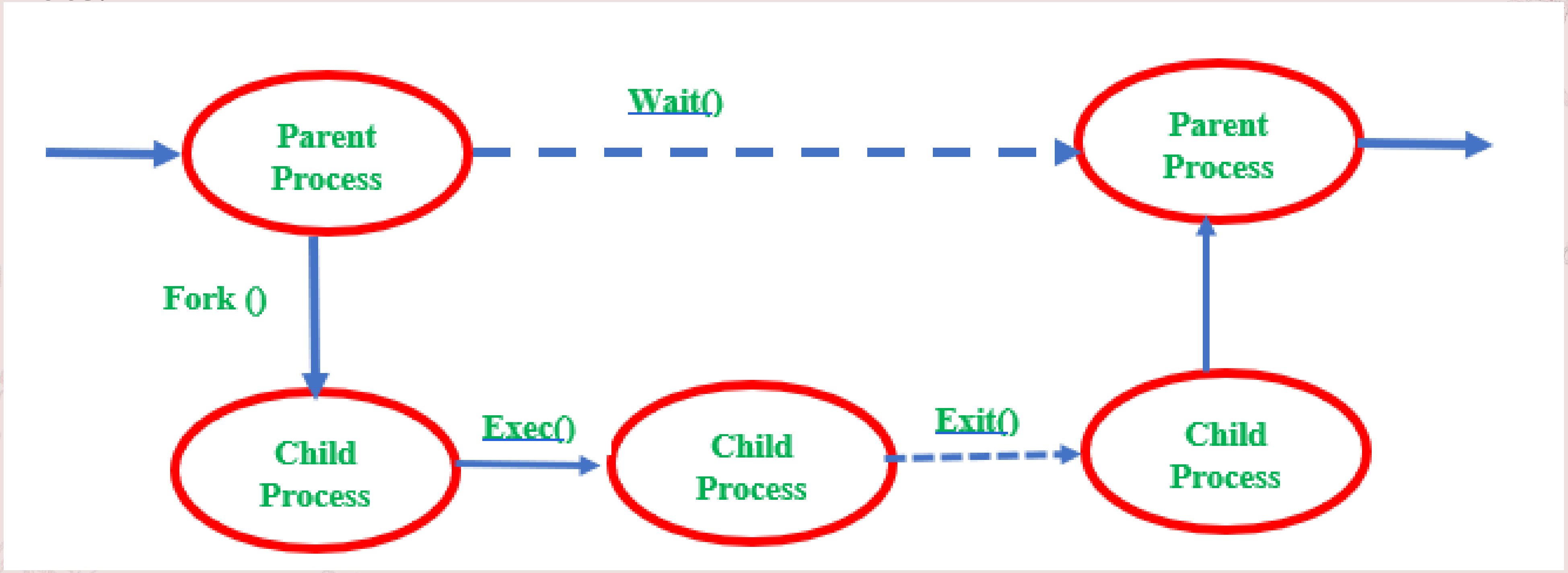
Process

Process	Thread
Process means any program is in execution.	Thread means a segment of a process.
The process takes more time to terminate.	The thread takes less time to terminate.
It takes more time for creation.	It takes less time for creation.
It also takes more time for context switching.	It takes less time for context switching.
The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
Multiprogramming holds the concepts of multi-process.	We don’t need multi programs in action for multiple threads because a single process consists of multiple threads.
The process is isolated.	Threads share memory.
The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.
The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents’ PCB, its own Thread Control Block, and Stack and common Address space.
A system call is involved in it.	No system call is involved, it is created using APIs.
The process does not share data with each other.	Threads share data with each other.

Process

Zombie Process :

A Zombie is a process that has completed its task but still, it shows an entry in a process table. The zombie process usually occurs in the child process. Very short time the process is a zombie. After the process has completed all of its tasks it reports the parent process that it has about to terminate.



Process

Orphan Process :

A child process that remains running even after its parent process is terminated or completed without waiting for the child process execution is called an orphan. A process becomes an orphan unintentionally. Some time intentionally become orphans due to the long-running time to complete the assigned task without user attention. The orphan process has controlling terminals.

Daemon process :

Daemon processes are started working when the system is bootstrapped and terminates only when the system is shut down. It does not have a controlling terminal. It always runs in the background.

Process

wait() and waitpid()

The wait() system call suspends execution of the calling process until one of its children terminates.
 The call wait(&status) is equivalent to: waitpid(-1, &status, 0);

The waitpid() system call suspends execution of the calling process until a child specified by pid argument has changed state. By default, waitpid() waits only for terminated children, but this behavior is modifiable via the options argument, as described below.

The value of pid can be:

< -1 meaning wait for any child process whose process group ID is equal to the absolute value of pid.   -1 meaning wait for any child process.
0 meaning wait for any child process whose process group ID is equal to that of the calling process.
> 0 meaning wait for the child whose process ID is equal to the value of pid.

Process

Zombie Process	Orphan Process	Daemon Process
A Zombie is a process that has completed its task but still, it shows an entry in a process table.	A child process that remains running even after its parent process is terminated or completed without waiting for the child process execution is called an orphan	A daemon process is a system-related process always running in the background.
Zombie process states always indicated by Z	The orphan process was created unknowingly due to a system crash.	Daemon process state indicated by ? in the field of *tty* column in the output
The zombie process has controlling terminals	The Orphan process has controlling terminals.	The daemon process does not have controlling terminals.
The zombie process treated as dead they are not used for system processing	An orphan process is a computer process even after their parent terminates init is become a parent and continue the remaining task.	A program that runs for a long time makes them as a daemon process and runs it in the background.
To remove the zombie process execute the kill command.	Terminate the Orphan process use the SIGHUP signal.	Daemon process only when system shutdown.

VM

Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory

A virtual memory is what its name indicates- it is an illusion of a memory that is larger than the real memory.

In this scheme, the User can load bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory
.
By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

VM

How does virtual memory work?

Virtual memory works by dividing the virtual address space used by a program into smaller units called pages. In this scheme, whenever some pages need to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering the main memory, the OS searches for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it has unlimited RAM.

What are the benefits of virtual memory?

- **Increased memory capacity:** It allows programs to use more memory than is physically available, enabling the execution of larger programs or multiple programs simultaneously.
- **Memory isolation:** Each program operates in its own virtual address space, ensuring that one program cannot access or modify the memory of another program.
- **Simplified memory management:** Virtual memory simplifies memory management for both the operating system and application developers by providing a uniform memory model.
- **Improved system stability:** Virtual memory helps prevent crashes and system instability by allowing the operating system to handle memory shortages and prioritize memory usage efficiently.

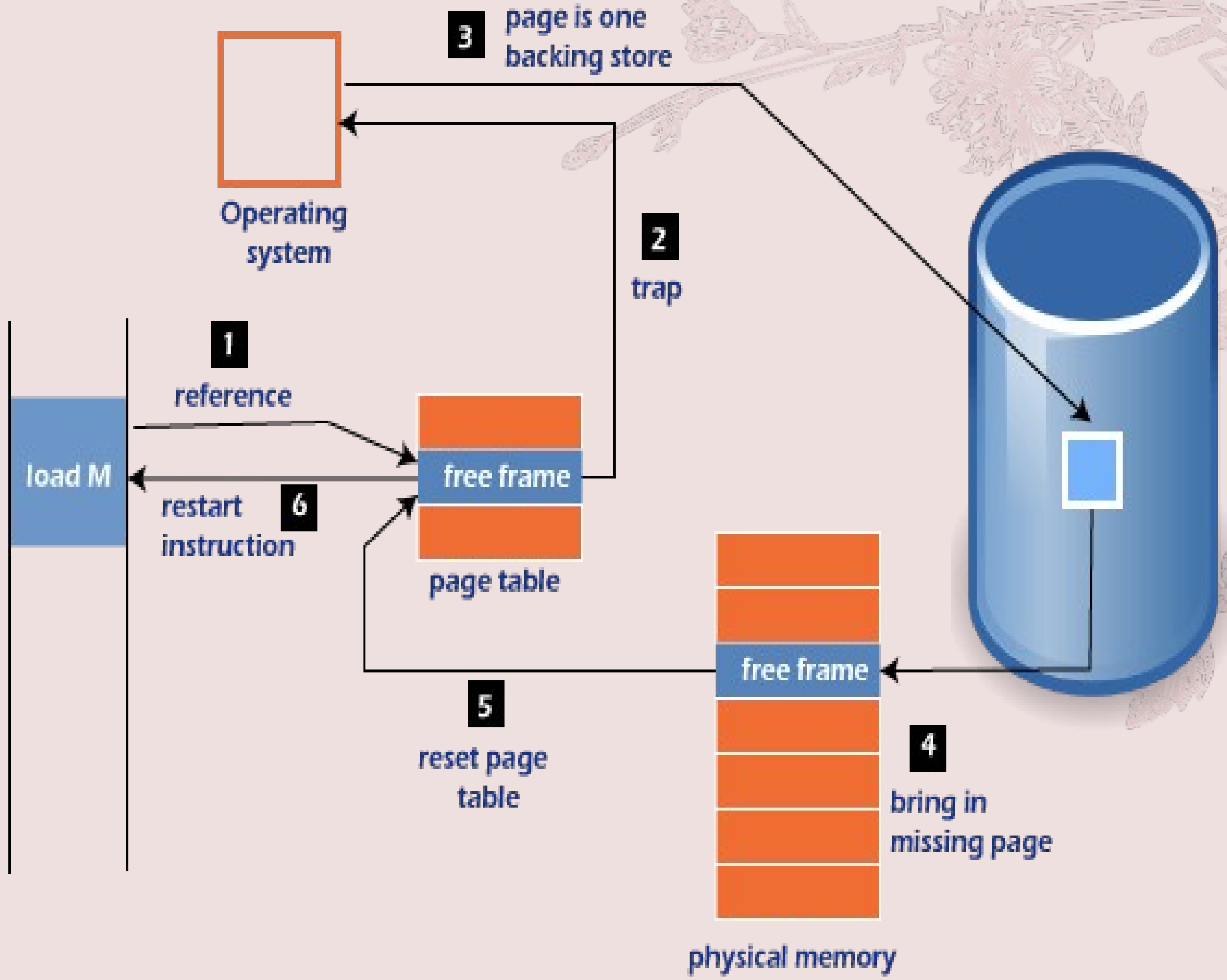
VM

What is a page fault?

The operating system manages a mapping between virtual addresses used by the program and physical addresses in the actual RAM or disk.

A page fault occurs when a program references a virtual memory page that is not currently resident in physical memory.

This happens when the required page has been paged out to disk or has not been accessed yet. When a page fault occurs, the operating system handles it by fetching the required page from the disk and updating the page tables to reflect the new mapping.



VM

Demand Paging

The process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging.

According to the concept of Virtual Memory, to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.

However, deciding, which pages need to be kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a process will require a particular page at a particular time.

Therefore, to overcome this problem, there is a concept called Demand Paging is introduced. It suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.
Whenever any page is referred to for the first time in the main memory, then that page will be found in the secondary memory.

VM

Page Replacement Algorithms

1. **First In First Out (FIFO):** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

2. **Optimal Page replacement:** In this algorithm, pages are replaced that would not be used for the longest duration of time in the future.

3. **Least Recently Used:** In this algorithm, the page will be replaced which is least recently used.

4. **Most Recently Used (MRU):** In this algorithm, the page will be replaced which has been used recently. Belady’s anomaly can occur in this algorithm.

OS- Deadlock

A ***deadlock*** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Deadlock can arise if the **following four conditions hold simultaneously (Necessary Conditions)**

- ***Mutual Exclusion:*** Two or more resources are non-shareable (Only one process can use at a time)
- ***Hold and Wait:*** A process is holding at least one resource and waiting for resources.
- ***No Preemption:*** A resource cannot be taken from a process unless the process releases the resource.
- ***Circular Wait:*** A set of processes waiting for each other in circular form.

OS- Deadlock

Methods for handling deadlock
There are three ways to handle deadlock

Deadlock prevention or avoidance:
Prevention:
The idea is to not let the system into a deadlock state. This system will make sure that above mentioned four conditions will not arise. These techniques are very costly so we use this in cases where our priority is making a system deadlock-free.
One can zoom into each category individually, Prevention is done by negating one of the above-mentioned necessary conditions for deadlock. Prevention can be done in four different ways:

1. Eliminate mutual exclusion
2. Solve hold and Wait
3. Allow preemption
4. Circular wait Solution

Avoidance:
Avoidance is kind of futuristic. By using the strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources that the process will need is known to us before the execution of the process. We use Banker's algorithm (Which is in turn a gift from Dijkstra) to avoid deadlock.
In prevention and avoidance, we get the correctness of data but performance decreases.

OS- Deadlock

Deadlock detection and recovery:

If Deadlock prevention or avoidance is not applied to the software then we can handle this by deadlock detection and recovery. which consist of two phases:

1. we examine the state of the process and check whether there is a deadlock or not in the system.
2. If found deadlock in the first phase then we apply the algorithm for recovery of the deadlock,

In Deadlock detection and recovery, we get the correctness of data but performance decreases.

3. Manual Intervention
4. Automatic Recovery
5. Recovery from Deadlock: **Process Termination**:
6. Abort all deadlocked processes
7. Abort one process at a time

Factors for choosing the termination order:

– The process’s priority
– Completion time and the progress made so far
– Resources consumed by the process
– Resources required to complete the process
– Number of processes to be terminated
– Process type (interactive or batch)

OS- Deadlock

Recovery from Deadlock: **Resource Preemption**:

Selecting a victim: Resource preemption involves choosing which resources and processes should be preempted to break the deadlock. The selection order aims to minimize the overall cost of recovery.

Rollback:
If a resource is preempted from a process, the process cannot continue its normal execution as it lacks the required resource. Rolling back the process to a safe state and restarting it is a common approach.

Starvation prevention:
To prevent resource starvation, it is essential to ensure that the same process is not always chosen as a victim. If victim selection is solely based on cost factors, one process might repeatedly lose its resources and never complete its designated task. To address this, it is advisable to limit the number of times a process can be chosen as a victim, including the number of rollbacks in the cost factor.

OS- Deadlock

Deadlock ignorance: If a deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take. we use the ostrich algorithm for deadlock ignorance.
In Deadlock, ignorance performance is better than the above two methods but the correctness of data.

Safe State:
A safe state can be defined as a state in which there is no deadlock. It is achievable if:
•If a process needs an unavailable resource, it may wait until the same has been released by a process to which it has already been allocated. if such a sequence does not exist, it is an unsafe state.
•All the requested resources are allocated to the process.

OS- Deadlock

Semaphores
Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization

The process of using Semaphores provides two operations: wait (P) and signal (V). The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.

They are used to enforce mutual exclusion, avoid race conditions, and implement synchronization between processes.

OS- Deadlock

Types of Semaphores

Counting Semaphores
These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate resource access, where the semaphore count is the number of available resources. If the resources are added, the semaphore count is automatically incremented and if the resources are removed, the count is decremented.

Binary Semaphores
The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when the semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores. This is also known as a mutex lock

Limitations :
1.One of the biggest limitations of semaphore is priority inversion.
2.Deadlock, suppose a process is trying to wake up another process that is not in a sleep state. Therefore, a deadlock may be blocked indefinitely.
3.The operating system has to keep track of all calls to wait and signal the semaphore.

OS- Deadlock

Advantages of Semaphores

- A simple and effective mechanism for process synchronization
- Supports coordination between multiple processes
- Provides a flexible and robust way to manage shared resources.
- It can be used to implement critical sections in a program.
- It can be used to avoid race conditions.

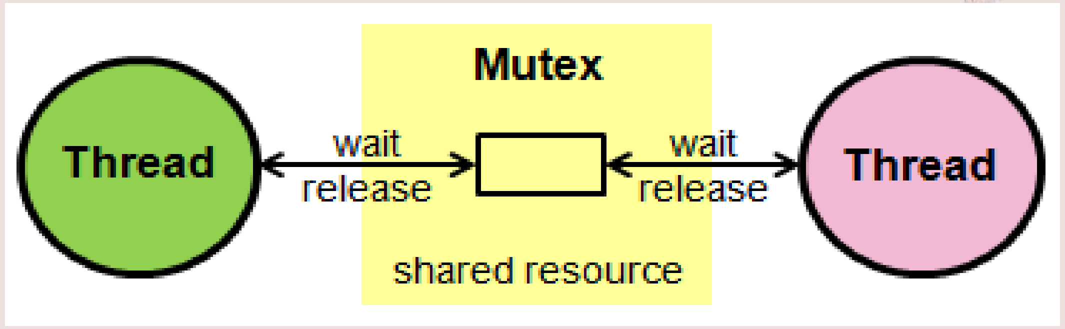
Disadvantages of Semaphores

- It Can lead to performance degradation due to overhead associated with wait and signal operations.
- Can result in deadlock if used incorrectly.
- It was proposed by Dijkstra in 1965 which is a very significant technique to manage concurrent processes by using a simple integer value, which is known as a semaphore. A semaphore is simply an integer variable that is shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.
- It can cause performance issues in a program if not used properly.
- It can be difficult to debug and maintain.
- It can be prone to race conditions and other synchronization problems if not used correctly.
- It can be vulnerable to certain types of attacks, such as denial of service attacks

OS- Deadlock

1. Process Synchronization- In a multiprogramming environment, where multiple processes are present, all the processes need to be synchronized so that incorrect output is not produced.
2. Race Condition- When two or more processes try to access the same shared variable, there is a possibility that it may produce an incorrect output which depends on the order in which access takes place.
3. Critical Section- A region that contains the shared variables.

What is Mutex in OS?
Mutex stands for mutual exclusion object. A mutex in OS is a program object used in computer programming that allows many program threads to alternately access the same resource, such as access to a file.
A mutex is a binary variable used to provide a locking mechanism. It offers mutual exclusion to a section of code that restricts only one thread to work on a code section at a given time.



OS-Deadlock

Working of Mutex

- Suppose a thread executes a code and has locked that region using a mutex.
- If the scheduler decides to perform context switching, all the threads ready to execute in the same region are unblocked.
- Out of all the threads available, only one will make it to the execution, but if it tries to access the piece of code already locked by the mutex, then this thread will go to sleep.
- Context switching will take place again and again, but no thread will be able to access the region that has been locked until the lock is released.
- Only the thread that has locked the region can unlock it as well.

This is how mutex ensures the mutual exclusion of threads in a multiprogramming environment.

OS-Deadlock

Advantages of Mutex
Since there is only one thread present in the critical section at a given time, there are no race conditions, and the data always remains consistent.
Only one thread will be able to access the critical section
Solves the race condition problem

Disadvantages of Mutex
If a thread is preempted or goes to sleep after obtaining a lock, then it may cause other threads to enter a state of starvation as they may not move forward.
It cannot be locked or unlocked other than the one thread that has acquired it.
It may lead to a busy waiting state, which wastes CPU time as the thread will check again and again for the lock to get released, hence wasting a lot of system calls.

OS-Deadlock

Components of Mutex Locks
The fundamental parts that constitute mutex locks include the lock itself, which is a synchronization mechanism preventing multiple threads from simultaneously accessing shared resources. These are several components of mutex locks:

- **Lock:** The core element that signals whether a resource is currently in use

- **Status Indicators:** Variables or flags indicating the state of the mutex, whether it's locked or unlocked

- **Thread Synchronization Mechanisms:** Protocols or methods ensuring orderly access to the shared resource by multiple threads

Different Types of Mutex Locks in OS
Several types of mutex locks are designed for specific scenarios.
Binary Mutex Locks: Basic on/off locks that either allow or prevent access

Recursive Mutex Locks: Permits a thread to relock a resource it has already locked

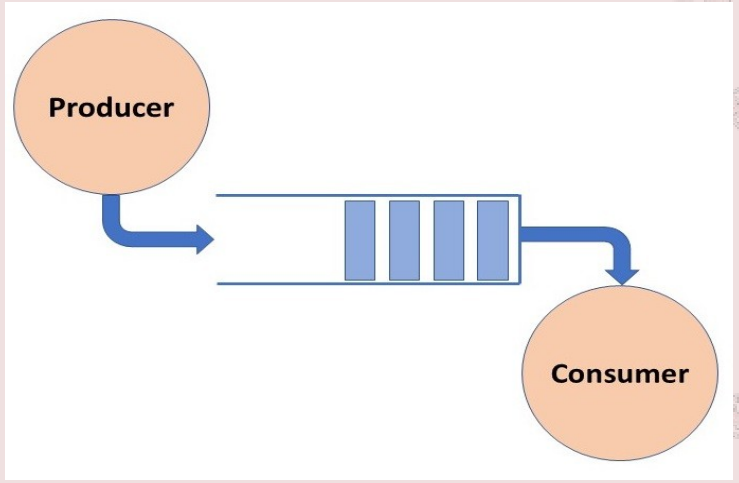
Adaptive Mutex Locks: Adjusts its behavior based on the current system conditions

OS-Deadlock

Deadlock	Starvation
All processes keep waiting for each other to complete and none get executed	High priority processes keep executing and low priority processes are blocked
Resources are blocked by the processes	Resources are continuously utilized by high priority processes
Necessary conditions Mutual Exclusion, Hold and Wait, No preemption, Circular Wait	Priorities are assigned to the processes
Also known as Circular wait	Also known as lived lock
It can be prevented by avoiding the necessary conditions for deadlock	It can be prevented by Aging

OS-Deadlock

The **Producer-Consumer problem** is a classic synchronization problem in operating systems.
The problem is defined as follows: there is a fixed-size buffer a Producer process, and a Consumer process.
The **Producer** process creates an item and adds it to the shared buffer. The **Consumer** process takes items out of the shared buffer and "consumes" them.



Certain conditions must be met by the Producer and the Consumer processes to have consistent data synchronization:

1. The Producer process must not produce an item if the shared buffer is full.
2. The Consumer process must not consume an item if the shared buffer is empty.
3. Access to the shared buffer must be **mutually exclusive**; this means that at any given instance, only one process should be able to access the shared buffer and make changes to it.

OS-Deadlock

The solution to the Producer-Consumer problem involves three *semaphore* variables.
Semaphore Full: Tracks the space filled by the Producer process. It is initialized with a value of 0 as the buffer will have 00 filled spaces at the beginning
Semaphore Empty: Tracks the space in the buffer. It is initially set to **buffer_size** as the whole buffer is empty at the beginning
semaphore Mutex: Used for mutual exclusion so that only one process can access the shared buffer at a time.
Using Signal and Wait operations on these semaphores, we can arrive at a solution.

```
void Consumer(){
   while(true){
      wait(Full);
      wait(mutex);
      consume();
      signal(mutex);
      signal(Empty)
   }
}
```

```
void Producer(){
   while(true){
      // Produce an item
      wait(Empty);
      wait(mutex);
      add();
      signal(mutex);
      signal(Full);
   }
}
```

Thank You