Unit 6: Computer Software (6 Hrs.)

Introduction

Software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system. They are the programs that enables a computer to perform a specific task. This includes application software such as word processor, which enables a user to perform a task, and a system software such as an operating system, which enables other software to run properly, by interfacing with hardware and with other software.

A computer system consists of hardware and software. The computer hardware cannot perform any task on its own. It needs to be instructed about the tasks to be performed. Software is a set of programs that instructs the computer about the tasks to be performed; hardware carries out these tasks. The components like monitor, keyboard, processor, and mouse, constitute the hardware.

Why does a computer need software?

Without software, a computer would be nothing more than a jumble of electronic components. The hardware components wouldn't be able to communicate or work together to perform any meaningful tasks. Software brings hardware to life by providing instructions, managing resources, enabling user interaction and performing specific tasks from word processing and web browsing to gaming and scientific calculations.

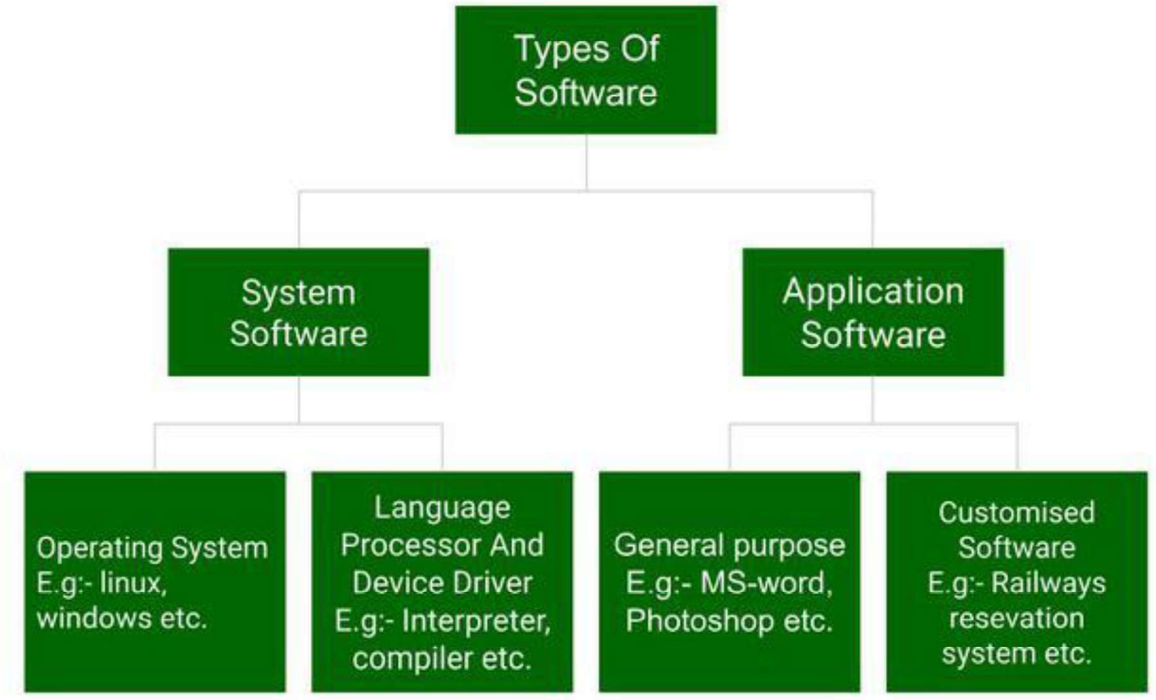
In conclusion, while hardware provides the foundation, software is the soul that breathes life into a computer. It's the driving force behind everything we do with computers, making them essential tools for work, communication, entertainment, and much more.

Types of Software

Software can be broadly classified in two categories:

i. System Software
ii. Application Software

System software consists of low-level programs that interact with the computer at very basic level. This includes operating systems, compilers, and utilities for managing resources. On the other hand, Application software includes database programs, word processors, and spreadsheets.


Types Of Software
System Software
Application Software
Operating System E.g:- linux, windows etc.
Language Processor And Device Driver E.g:- Interpreter, compiler etc.
General purpose E.g:- MS-word, Photoshop etc.
Customised Software E.g:- Railways resevation system etc.

System Software

- **Purpose:** Manages and controls the computer's hardware and other resources. Acts as the foundation upon which application software runs.
- **Function:**
    - Manages memory, storage, and processing power.
    - Provides an interface between hardware and application software.
    - Handles input/output operations (connecting to peripherals).
    - Provides security and error handling.
- **Examples:** Operating Systems (Windows, macOS, Linux), Device Drivers, Device Managers, File Systems.
- **Characteristics:**
    - Pre-installed on the computer.
    - Runs in the background, mostly invisible to the user.
    - Written in low-level languages for performance and direct hardware interaction.
    - Not user-specific, provides general functionalities.

The system software can be sub-divided as follows:

a. Operating System
b. Translator (language processor)
c. Utility Software

Operating System

- Considered the **heart of any computer system**.
- Acts as an **interface** between the user, hardware, and application software.
- **Manages and controls** vital resources like memory, storage, processing power, and input/output operations.
- Provides a platform for running application software.
- Examples: Windows, macOS, Linux, Android, iOS.

Key functions:

- Booting the system.
- Managing files and folders.
- Launching applications.
- Handling security and permissions.
- Providing a user interface (graphical or command-line).
- Communicating with hardware devices through drivers.

Translators

Computer only understand the program written in machine language. Hence the program written in other language must be translated into machine language program before executing them. Translator is a computer programs that convert the program written in other language into an equivalent machine language program before executing them. Programs written in other language is called source program and the program that is obtained after converting into machine code is called object program. Some of the translating programs are as follows:

- **Compilers:** Translate high-level languages (C++, Java, Python) into machine code understood by the computer. This machine code is then executable by the system.
- **Interpreters:** Execute code line by line, translating each line from a high-level language into machine code on the fly. They don't create separate executable files but offer faster development cycles.
- **Assemblers:** Convert assembly language (closer to machine code than high-level languages) into machine code directly usable by the processor.

Utility Software

- A broad category encompassing various tools for **system maintenance, optimization, and specific tasks**.
- Focuses on improving system performance, managing files, enhancing security, and providing valuable functionalities.
- Examples: Anti-virus software, disk defragmenters, file compression tools, backup software, system diagnostics tools.

Key functions:

- Optimizing storage space and system performance.
- Detecting and removing malware and viruses.
- Managing and organizing files and folders.
- Backing up and restoring data.
- Automating repetitive tasks.

Application Software

- **Purpose:** Helps users perform specific tasks.
- **Function:**
  - Wide range of functionalities - creating documents, editing photos, playing games, browsing the web, etc.
  - User-friendly interface for direct interaction.
  - Tailored to specific needs and interests.
- **Characteristics:**
  - Installed by the user based on their needs.
  - Runs in the foreground, directly controlled by the user.
  - Written in both high-level and low-level languages depending on complexity.
  - User-specific and diverse in scope and function.

Application software is written for different kinds of applications – graphics, word, processors, media players, database applications, telecommunication, accounting purposes etc.

The examples of some application software are:

Word processing package: MS-Word, Word perfect

Spread sheet package: MS-Excel, Lotus 1-2-3

Database package: MS Access, Oracle, FoxPro

Engineering Package: CAD/CAM

There are two types of application software: -

Customized or Tailored Software:
Customized software is the software designed and developed specifically to meet the unique needs and requirements of an individual or organization. For example, Software for air traffic controls system, software for billing school fees etc.

Advantages:

- It is easy to modify
- It is very cheap

Disadvantages:

Different problems may arise frequently because:

- It is partially developed
- It is partially tested.

Packaged Software:
it's mass-produced software designed for general use by a broad audience. It is also called universal software as it can be used by users and organizations all over the world. Example of this type of software includes; word processing package, spreadsheet package, database package etc.

Advantages:

- It gives quick result
- It is user friendly and easy to run.
- Using this package we can save time and money.
- It should have menu driven facilities.
- It is designed for general purpose.
- Multiple software distributed or sold in one package.

Disadvantages

- It has poor documentation
- There is possibility of coming up of new version of package. Every time, it is difficult to keep track and bear buying new version.
- It is designed to meet the need of number of different users, so it may not be exactly suitable for one's need.

Differences between System Software and Application Software

Key differences:

Feature	System Software	Application Software
Purpose	Manage hardware and resources	Perform specific tasks
Function	Background	Foreground
User interaction	Generally, no interaction	User-friendly interface
Installation	Pre-installed	User-installed
Programming languages	Low-level	High-level and low-level
Specificity	General	User-specific

System Software	Application Software
i. System software is a group of programs that direct the internal operations of computer system such as controlling I/O devices, managing the storage area within the computer etc.	The software that is written by the user to solve a specific user-oriented problem using the computer is known as application software.
ii. Generally, users do not interact with system software	Generally, users interact with application software.
iii. System software runs independently.	Application can't run without the present of the system software.
iv. System software perform several tasks.	Application software perform specific task.
v. It is of three types	It is of two types.

a. Operating system b. Language processor c. Utility software	. Tailored software a. Packaged software
vi. Example: Windows XP, Linux	Example: Adobe, MS Word

Software Acquisition

Software acquisition refers to the process of obtaining software for an organization or individual. The user may have to purchase the software, can download for free from the internet, or can get it bundled along with the hardware. Nowadays with the advent of Cloud computing, many applications software are also available on the cloud for use through the internet, e.g. Google Drive.

The different ways in which the software are made available are as follows:

1. Retail Software:

- Purchased at physical stores or online retailers, typically with a boxed copy and license.
- Examples: Microsoft Office, Adobe Photoshop.

2. OEM Software:

- OEM (Original Equipment Manufacturer) software refers to applications that have been sold to hardware and software manufacturers wholesale for the purpose of bundling with existing offerings.
- Pre-installed on new computers by the manufacturer.
- Often limited versions or trial editions.
- Examples: Windows operating systems on new PCs.

3. Demo Software:

- Non-functional or limited-functionality versions offered for trial purposes.
- Allows users to test the software before purchasing.
- Examples: Trial versions of games or productivity software.

4. Shareware:

- Distributed freely for a limited time or with limited features.
- Encourages users to purchase a full license for continued use.
- Examples: WinRAR, WinZip.

5. Freeware:

- Available at no cost, with full functionality.
- May be supported by donations, advertising, or bundled with other software.
- Examples: VLC media player, Audacity.

6. Public Domain Software:

- No copyright restrictions, allowing free use, modification, and distribution.
- Often older software or software donated by creators.
- Examples: Classic games like Tetris or Doom.

7. Open-Source Software:

- Source code is freely available for anyone to inspect, modify, and redistribute.
- Encourages community collaboration and development.
- Examples: Linux operating system, Mozilla Firefox web browser.

let's delve into the world of programming languages, exploring high-level and low-level languages.

Programming Languages:

Programming languages act as a bridge between humans and computers. They provide a set of instructions that the computer can understand and execute to perform specific tasks. These instructions are written in a specific syntax (grammar) with keywords and rules that the programming language defines.

High-Level Languages: Human-Centric Communication

- **Human-Friendly:** High-level languages resemble natural languages like English. They use keywords and commands that are easier for programmers to understand and write.
- **Abstraction:** They provide a layer of abstraction, meaning programmers don't need to worry about the underlying hardware details. You can instruct the computer to "print a message" without diving into how the computer displays it on the screen.
- **Portability:** High-level languages are portable, meaning the same code can run on different computers with a compiler (a program that translates the code into machine code the specific computer understands). Examples of high-level languages include Python, Java, C++, JavaScript, etc.

Low-Level Languages: Talking to the Machine

- **Machine-Friendly:** Low-level languages are closer to the machine's native language. They offer more control over hardware but are less intuitive for humans to understand.

- **Less Abstraction:** Programmers need a deeper understanding of the computer's architecture and memory management when using low-level languages.
- **Non-Portable:** Low-level languages are typically specific to a particular computer architecture and may not run on different machines without modification.

Generations of Low-Level Languages

Machine Language (1st Generation): Raw Instructions

- The most basic and fundamental level.
- Instructions are represented in binary code (0s and 1s) that the computer directly understands.
- Extremely difficult for humans to read, write, and understand.
- Not portable – specific to the underlying hardware.

Assembly Language (2nd Generation): A Step Up from Binary

- Uses mnemonics (abbreviated codes) that are easier to remember than binary codes.
- Assembly language instructions still correspond to machine code instructions.
- Requires an assembler program to translate assembly code into machine code.
- Offers more control over hardware compared to high-level languages but less portable.

Choosing the Right Tool for the Job

- High-level languages are generally preferred for most development tasks due to their ease of use, readability, and portability.
- Low-level languages are still used in specific scenarios where:
    - Fine-grained control over hardware is essential (device drivers, operating system kernels)
    - Performance is critical (game development)

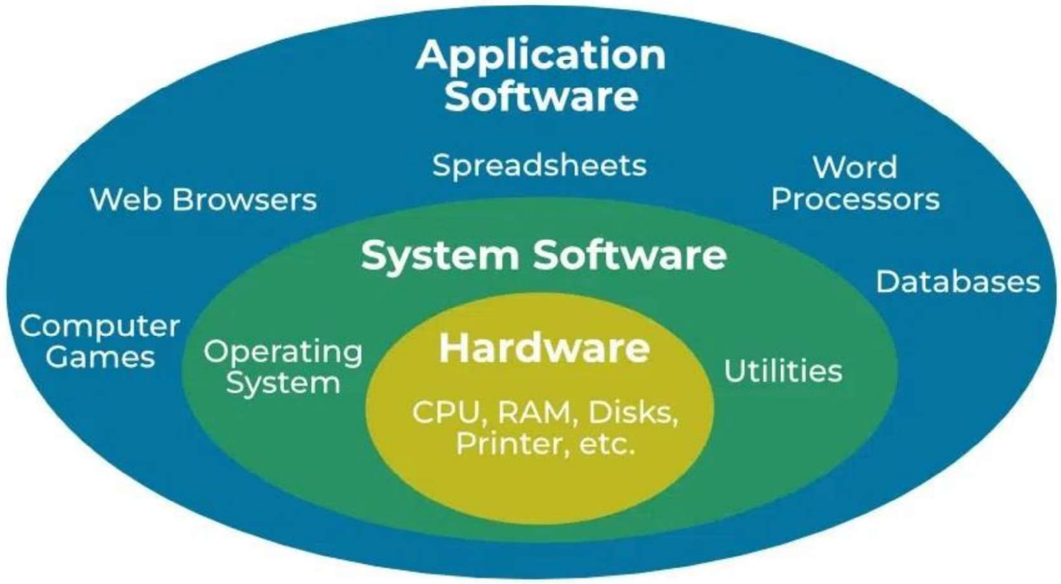
Operating System

Introduction

Definition

An operating system is system software, which is set of specialized programs that are used to control the resources of a computer system.

An operating system (OS) is the **software** that acts as the **maestro** of computer system, managing all its resources and facilitating communication between user and the hardware.



Objectives of Operating System

1) An operating system (OS) has several objectives that all work together to make using a computer easier and more efficient. OS has two main objectives:
   To make the computer system convenient and easy to use, for the user, and
2) To use the computer hardware in an efficient way, by handling the details of the operations of the hardware.

Types of OS

1. Single User OS:

- Designed for one user at a time.
- Common on early personal computers and some embedded systems.
- Simpler and less resource-intensive compared to multi-user systems.
- Example: MS-DOS (early versions). In DOS we cannot run another program at same time. For this we will have to close the first program, only then we would be able to work on another program or software.

2. Multi-User OS:

- Allows multiple users to access the system concurrently.
- Manages resources (CPU, memory) to ensure fair and efficient sharing among users.
- Requires security measures to isolate user accounts and protect data.
- Examples: Windows Server, Linux, macOS

3. Multitasking OS:

- Enables a single user to run multiple programs simultaneously.
- Provides the illusion of multiple programs running at the same time by rapidly switching between them.
- Requires efficient memory management to keep track of running programs and their data.
- Most modern operating systems are multitasking.
- Examples: Windows, macOS, Android

4. Multiprocessing OS:

- Designed for computers with multiple central processing units (CPUs).
- Can distribute tasks among multiple CPUs for faster overall processing.
- Requires the OS to manage task scheduling and synchronization across CPUs.
- Used in high-performance computing systems and servers.
- Examples: Windows Server (with multiple CPUs), Linux (with SMP support)

5. Real-Time OS (RTOS):

- A real time operating system (RTOS) is a type of operating system that is designed to control and manage real-time systems.
- Focuses on guaranteeing predictable response times to events.
- Used in systems where timely response is crucial, like industrial control systems, medical devices, and robotics.
- Prioritizes tasks based on their time-critical nature.
- Often lightweight and designed for specific applications.
- Examples: FreeRTOS: FreeRTOS is an open-source real-time kernel that offers a small footprint, portability, and rich set of features.

  There are two types of real time OS
    1.) Hard Real time OS
    2.) Soft Real time OS

1. Hard Real-Time OS (HRT OS):

- **Strict Deadlines:** These systems are designed to guarantee that critical tasks are completed within their deadlines. Missing a deadline in a Hard Real-Time system can have severe consequences, potentially causing system failure or safety hazards.
- **Resource Management:** HRT OS carefully manages resources to meet deadlines. It prioritizes tasks with the most critical deadlines and may even reject tasks if there's no guarantee they can be completed on time.

- **Applications:** HRT OS is used in mission-critical systems where even a slight delay can be catastrophic. Examples include:
    - Flight control systems in airplanes
    - Anti-lock braking systems (ABS) in cars
    - Industrial robot controllers
    - Life support systems in hospitals

2. Soft Real-Time OS (SRT OS):

- **Relaxed Deadlines:** SRT OS aims to meet deadlines, but occasional misses are tolerable. While performance degrades when deadlines are missed, it doesn't necessarily lead to system failure or severe consequences.
- **Flexibility:** SRT OS offers more flexibility in resource management. It can accommodate a wider range of tasks, even if some deadlines are occasionally missed.
- **Applications:** SRT OS is suitable for systems where timeliness is important, but missing deadlines doesn't have disastrous consequences. Examples include:
    - Multimedia playback systems (video streaming)
    - Virtual reality applications
    - Voice over IP (VoIP) systems
    - Stock trading applications

6. Embedded OS:

- Designed for specific devices with limited resources (memory, processing power).
- Often tailored to the specific functionality of the device.
- Found in devices like smartphones, printers, smartwatches, and internet of things (IoT) devices.
- Prioritizes low memory footprint and efficient power usage.
- Examples: Android (on some devices), iOS, FreeRTOS (on some devices)

Functions of OS

- **Resource Management:** The OS oversees and allocates resources like memory, storage, and processing power to ensure smooth operation of various programs.
- **Interface Management:** It acts as the **interpreter** between user and the computer hardware, providing a user-friendly interface (like the desktop or command prompt) to interact with the system.
- **Program Execution:** The OS loads and runs applications, ensuring they have the necessary resources and follow specific rules for safe and efficient execution.

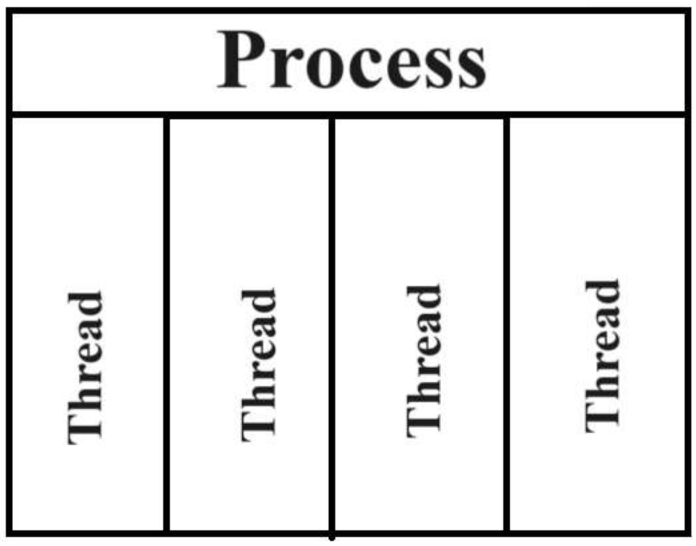
- **Device Management:** It controls and communicates with various devices connected to the computer, like printers, scanners, and keyboards.

Security: The OS implements security measures to protect the computer system from unauthorized access and harmful software.

Process Management

Process
A process is an instance of a computer program in execution.
A process will need certain resources such as CPU time, memory, files, and I/O devices to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

Thread
A thread is the unit of execution within a process. A process can have anywhere from just one thread to many threads.



Process state

As process executes, it changes state. The state of a process is defined in part by the curret activity of the process.

Each process may be in one of the following states:

New ➔ A process is being created.

Running ➔ Instruction are being executed.

Waiting ➔ The process is waiting for some event to occur. (Such as I/O completion or reception of a signal). In waiting state, also referred to as the blocked state, the process is essentially paused and unable to utilize the CPU.

Ready ➔ The process is waiting to be assigned to a processor.

Terminated ➔ The process has finished execution.

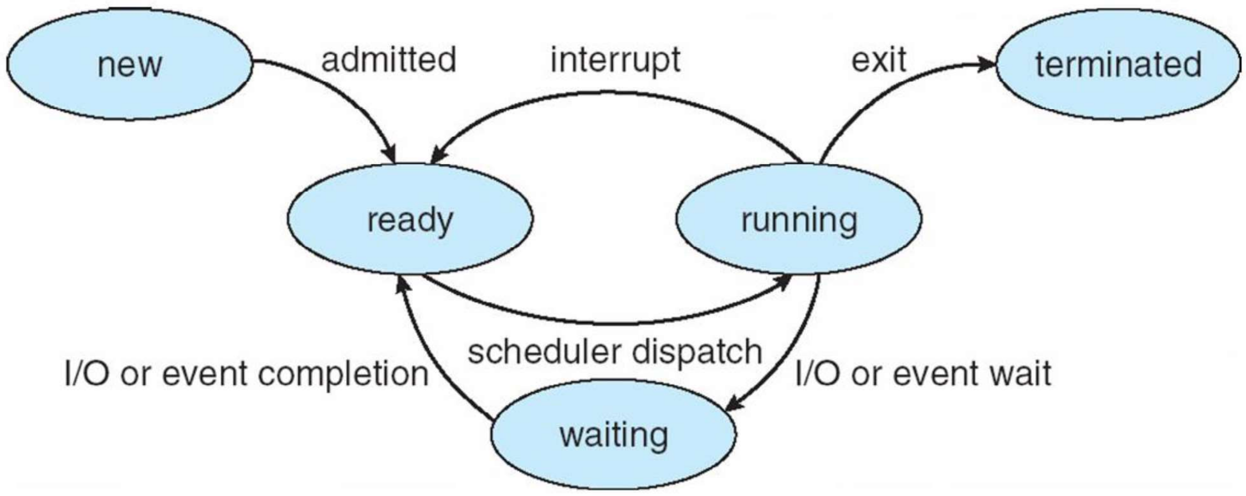


Figure 4: Diagram of process state

CPU Scheduling

CPU scheduling is the fundamental operating system process that decides which program (process) gets to use the central processing unit (CPU) at a given time. Imagine a busy restaurant with one chef (CPU). Scheduling determines which customer (process) gets served (uses the CPU) next.

- **Scheduling Algorithms:** There are various CPU scheduling algorithms, each with its own advantages and trade-offs. Some common algorithms include:
  - **First Come First Served (FCFS):** Processes are served in the order they arrive. This is fair but can lead to starvation for shorter processes if a long process arrives first.
  - **Shortest Job First (SJF):** The process with the shortest execution time is chosen next. This minimizes average waiting time but requires knowing the execution time upfront, which is often not practical.
  - **Priority Scheduling:** Processes are assigned priorities, and higher priority processes are served first. This is useful for ensuring critical tasks get CPU time promptly.

    - **Round Robin (RR):** Processes are allocated CPU time in short slices (time quanta). When a slice ends, the process is preempted (paused), and another process gets a turn. This provides a balance between fairness and responsiveness.
- **Goals of CPU Scheduling:** The ideal CPU scheduling algorithm aims to achieve several goals:
    - **Maximize CPU Utilization:** Keep the CPU busy as much as possible to avoid wasted cycles.
    - **Improve Throughput:** The number of processes completed per unit time.
    - **Minimize Waiting Time:** The time a process spends waiting for the CPU.
    - **Minimize Response Time:** The time it takes for a process to start responding after submitting a request.
    - **Ensure Fairness:** All processes should have a chance to use the CPU without indefinite starvation.

Memory Management

Memory management is a crucial function of an operating system (OS) that handles the allocation, deallocation, and organization of a computer's main memory (RAM) to the processes and their data at the time of their execution. RAM is essential for processes to run, but it's a limited resource. Memory management ensures efficient and secure utilization of RAM by multiple programs simultaneously.

Memory management also performs following activities:

- Upgrading the performance of the computer system
- Enabling the execution of multiple processes at the same time
- Sharing the same memory space among different processes.

Key aspects of effective memory management:

- **Allocation:** The OS keeps track of free memory locations and assigns them to processes as needed. This ensures each process has sufficient space to store data and instructions for execution.
- **Deallocation:** When a process finishes or no longer requires allocated memory, the OS reclaims that memory, making it available for other processes.
- **Protection:** Memory management prevents processes from interfering with each other's memory space. This safeguards data integrity and system stability.
- **Provision to share the information:** An ideal memory management system must facilitates sharing of data among multiple processes.
- **Virtual Memory:** Modern OSes employ virtual memory, a technique that creates the illusion of having more RAM than physically available. It partitions the hard disk into a

designated space and utilizes it as an extension of RAM. Programs are divided into fixed-size blocks (pages) and loaded into RAM as needed. When a program accesses data from a non-resident page, the OS retrieves it from the disk and swaps it with a less frequently used page in RAM. This technique allows larger programs to run than the physical RAM capacity.

Memory Allocation

Memory allocation techniques in operating systems deal with how the OS assigns memory blocks to various processes.
There are different memory management techniques used by operating systems, including:

- **Paging:** Divides both physical memory (RAM) and logical memory (program address space) into fixed-size blocks for allocation and management. The divided fixed sized blocks of physical memory are called frames and of logical memory is called pages. Generally, pages are of sizes varying from 1 KB to 8 KB. When a process is executed, its pages are loaded into the frames.
- **Segmentation:** Splits a program's logical address space into variable-sized segments based on functionality (code, data, etc.). Segments are then loaded into contiguous memory blocks. This approach allows for better memory protection and management for different program sections.

File Management

File is a collection of related information, has a name, and is stored on a secondary storage. File management in an operating system (OS) is all about organizing, storing, and manipulating files on a computer's storage devices.

key aspects of file management:

- **File System:** The foundation of file management is the **file system**. It's a method used by the OS to organize files and folders on storage devices like hard drives or solid-state drives. The file system keeps track of file attributes like name, size, type (document, image, etc.), creation/modification time, and location within the storage device. Common file system examples include NTFS (Windows), ext (Linux), and HFS+ (macOS).
- **File Organization:** File systems typically employ a **hierarchical structure** using directories (also called folders). Folders can contain files and other subfolders, creating a tree-like organization. This allows you to categorize and group related files for better accessibility and management.
- **File Operations:** The OS provides various functionalities to interact with files. These include:

    - **Creating and Deleting Files:** You can create new files and delete unwanted ones.
    - **Renaming and Moving Files:** You can change file names and move them between folders for organization.
    - **Copying and Pasting Files:** You can duplicate files and folders within or across storage devices.
    - **Opening and Editing Files:** You can launch applications to open and edit files based on their type.
- **File Permissions:** File systems can implement access control mechanisms using **permissions**. These determine who (users or groups) can access, modify, or delete specific files. This helps ensure data security and privacy.
- **File Searching:** Locating specific files can be done using the OS's built-in search functionality. You can search by filename, file type, content keywords, or other criteria.

Device Management

Several peripheral devices like mouse, hard disk, printer, plotter etc. are connected to the computer. OS manages and controls the devices attached to the computer. Device management is responsible for managing all the hardware devices of the computer system. It ensures these devices function properly and communicate seamlessly with the software used in computer system.

Key aspects of device management includes:

- **Device Discovery and Identification:** When you connect a device (printer, scanner, webcam, etc.) to your computer, the OS initiates the device management process. It first recognizes the device and gathers information about its type, manufacturer, and model. This allows the OS to load the appropriate drivers for it.
- **Device Drivers:** Device drivers act as translators between the OS and the hardware. They are specialized software programs that understand the specific communication protocol of a device and convert generic OS instructions into commands the device can comprehend. Without proper drivers, a device might not function at all or operate erratically.
- **Resource Allocation:** The OS manages the allocation of system resources to devices. This includes things like memory, interrupt handling (handling signals from devices requiring attention), and I/O (input/output) channels for data transfer between the device and the system.
- **Device Configuration and Monitoring:** Device management allows you to configure settings for various devices. You can adjust printing preferences, microphone volume, or webcam resolution through the OS interface. The OS also monitors device status, detecting any errors or malfunctions and potentially offering troubleshooting options.

- **Plug and Play (PnP):** Modern operating systems often support Plug and Play (PnP) functionality. This allows automatic device configuration and driver installation when you connect a compatible device. PnP simplifies device management for users.
- **Device Sharing:** Device management can facilitate sharing of certain devices, like printers, across a network. This allows multiple users on the network to access and utilize the shared device.

Protection and Security

While the terms "protection" and "security" are often used interchangeably, they have distinct meanings in the context of operating systems (OS).

Protection:

- **Focus:** Deals with **internal threats** within the system, ensuring that various programs and processes running on the system do not interfere with each other's resources or corrupt data.
- **Mechanisms:**
    - **Memory Management:** Isolates the memory space of each process, preventing one process from accessing or modifying the memory of another.
    - **File System Permissions:** Controls access to files and folders, restricting unauthorized users or processes from reading, modifying, or deleting them.
    - **Resource Management:** Allocates system resources like CPU time, I/O channels, and devices fairly and securely among different processes.

Security:

- **Focus:** Protects the system from **external threats** like unauthorized access, malware attacks, data breaches, and other malicious activities.
- **Mechanisms:**
    - **User Authentication:** Requires users to provide valid credentials (username and password) before accessing the system, preventing unauthorized login attempts.
    - **Authorization:** Defines access levels for different users and groups, controlling what actions they can perform within the system.
    - **Firewalls:** Filter incoming and outgoing network traffic, blocking suspicious or unauthorized connections that could pose a security risk.
    - **Anti-malware software:** Detects and removes malware like viruses, worms, and trojans that can harm the system or steal data.
    - **Encryption:** Scrambles data to protect its confidentiality, making it unreadable to unauthorized individuals even if intercepted.

User Interface

The primary goal of OS is to make the computer convenient for use by its user. It should allow user to easily access and communicate with the applications and the hardware. The user interface (UI) in an operating system (OS) act as a bridge between the user and the underlying functionalities of the computer. It provides a visual or textual environment for users to interact with the OS, manage files and applications, and issue commands.

Types of UIs in Operating Systems:

- **Command-Line Interface (CLI):** This traditional interface uses text-based commands for users to interact with the system. While powerful for experienced users, CLIs can be challenging to learn and use for beginners. MS-DOS and Linux shell are examples of command line mode of interface.
- **Graphical User Interface (GUI):** The most common type today, GUIs employ visual elements like windows, icons, menus, and buttons to provide a more intuitive and user-friendly experience. Modern GUIs often incorporate multimedia elements like graphics and animations to enhance the interaction. Windows 7 and Mac OS 10 are examples of graphical mode of interface. GUI interface for the Linux OS also exist like the GNU object model environment (GNOME).
- **Touch-based Interfaces:** With the rise of mobile devices and tablets, touch-based UIs have become increasingly prevalent. These interfaces rely on gestures like tapping, swiping, and pinching to interact with the system, optimized for use with touchscreens.

Examples of Operating Systems

MS-DOS

- MS-DOS was the first widely-installed CUI operating system for PCs in 1980s.
- MS-DOS is a command line user interface operating system. This means that the user has to type single line commands through the command interface. So, user has to remember the different commands and their syntax.
- MS-DOS is easy to load and install. It neither requires much memory for the operating system, nor a very powerful computer to run on.
- It is a single-user and single-tasking operating system for the PC. Only one user can use it and only one task can be executed, at a given point of time. Also, it does not have a built-in support for networking.
- MS-DOS is a 16-bit OS, meaning thereby that it can send or receive 16 bits of data at a time and can process 16 bits of data. It is not able to take the advantage of 32-bit processors.

Windows Family of OS

Early Windows (1985-1992):

- Windows 1.0 (1985): Pioneered a graphical interface for MS-DOS, marking the beginning of the Windows journey.
- Windows 2.0 (1987): Introduced overlapping windows and enhanced graphics, building upon the foundation laid by Windows 1.0.
- Windows 3.0 & 3.1 (1990-1992): Further refined the graphical user experience with features like Program Manager and File Manager, laying the groundwork for future advancements.

The Rise of Consumer-Centric Windows (1995-2000):

- Windows 95 (1995): A game-changer, introducing a user-friendly interface with the iconic Start menu and taskbar, propelling Windows to mainstream adoption.
- Windows 98 & Me (1998-2000): Built upon the success of Windows 95, offering improved stability, multimedia capabilities, and USB device support.

Windows NT: A Focus on Business (1993-2000):

- Windows NT (New Technology): Launched in 1993, targeted business users with its emphasis on improved security and stability.
- Windows 2000 (2000): Merged the consumer and business lines, offering a robust and feature-rich platform for diverse users.

The Era of Widespread Adoption (2001-2020):

- Windows XP (2001): Became a global phenomenon, renowned for its user-friendly interface, enhanced multimedia features, and exceptional stability.
- Windows Vista (2007): Aimed to elevate security and user experience, but faced criticism due to performance and resource demands.
- Windows 7 (2009): Addressed the shortcomings of Vista, offering a streamlined interface, improved performance, and a superior user experience.

The Modern Era of Windows (2012-present):

- Windows 8 & 8.1 (2012-2015): Introduced a touch-centric interface with the Start screen and Modern UI apps, catering to the growing tablet market.
- Windows 10 (2015-present): Shifted towards a "Windows as a Service" model, delivering continuous updates and improvements. It offered a unified interface across devices, introduced Cortana (virtual assistant), and provided the Microsoft Store for applications. Additional features like virtual desktops, the Action Center, and the Edge browser further enriched the experience.

- Windows 11 (2021-present): The latest iteration boasts a redesigned Start menu and taskbar, alongside innovative Snap layouts for multitasking. It emphasizes a more streamlined and centered user experience, improved performance, and enhanced features for gamers.

Linux OS

Linux is a free open-source computer operating system, initially developed for **Intel x86**-based personal computers. It has been subsequently ported to many other hardware platforms.

Linus Torvalds was a student in Helsinki, Finland, in 1991, when he started a project: writing his operating system **kernel**. He also collected together and/or developed the other essential ingredients required to construct an entire **operating system** with his kernel at the center. Soon, this became known as the **Linux** kernel.

In 1992, Linux was re-licensed using the General Public License (GPL) by GNU (a project of the Free Software Foundation or FSF, which promotes freely available software), which made it possible to build a worldwide community of developers. By combining the kernel with other system components from the GNU project, numerous other developers created complete systems called Linux Distributions in the mid-90's.

- Linux borrows heavily from the **UNIX** operating system, with which its creators were well-versed.
- Linux accesses many features and services through files and file-like objects.
- Linux is a fully multi-tasking, multi-user operating system, with built-in networking and service processes known as daemons.
- **GNOME** is a popular desktop environment and graphical user interface that runs on top of the Linux operating system.

Assignment:

Apple macOS
Android OS
IOS