CSC2210 OPERATING SYSTEM I

i. Introduction to Operating Systems:

a. Definition

OS is defined as a program or suite of programs that controls the entire operation of the computer.

Can also be defined as the software responsible for allocating and managing system resources

System Resources can be a memory, processor time, disk space, printer, scanner etc

b. Evolution and historical development.

i. 1940s - 1950s: Early Computer Systems:

Characteristics: The earliest electronic computers did not have operating systems. Programs were loaded manually, and each task required a dedicated machine.

Example: ENIAC (Electronic Numerical Integrator and Computer).

ii. 1950s - 1960s: Batch Processing Systems:

Characteristics: Batch processing systems allow users to submit jobs in batches. Operating systems were developed to automate job scheduling and resource allocation.

Example: UNIVAC I with the UNIVAC I Operating System (UNIVAC I OS).

iii. 1960s - 1970s: Time-Sharing Systems:

Characteristics: Time-sharing systems allow multiple users to interact with the computer simultaneously, providing each user with a share of the computer's resources.

Example: CTSS (Compatible Time-Sharing System), MULTICS.

iv. 1970s: Rise of Microcomputers and Single-User Systems:

Characteristics: The emergence of microprocessors led to the development of single-user operating systems for personal computers.

Example: CP/M (Control Program for Microcomputers), Apple DOS.

v. 1980s: Graphical User Interfaces (GUI) and Networking:

Characteristics: GUIs made computers more user-friendly. Networking capabilities allowed computers to communicate and share resources.

Examples: MS-DOS, Apple Macintosh System Software, UNIX.

vi. 1990s: Client-Server Computing and Windows Dominance:

Characteristics: Client-server architectures became popular. Microsoft Windows gained widespread adoption in personal computing.

Examples: Windows 95/98, Windows NT, Linux.

vii. 2000s: Mobile Operating Systems and Virtualization:

Characteristics: The rise of mobile computing led to the development of mobile operating systems. Virtualization technologies became more prominent for server efficiency.

Examples: Windows XP, macOS, Linux, Android, iOS.

viii. 2010s - Present: Cloud Computing and Modern OS Features:

Characteristics: Cloud computing has become integral. Modern operating systems emphasize security, virtualization, and support for diverse hardware.

Examples: Windows 10, macOS, Linux distributions, Android, iOS.

c. Functions/Roles of Operating System

i. File Management

Operating systems are responsible for managing the files on a computer. This includes creating, opening, closing, and deleting files. The operating system is also responsible for organizing the files on the disk.

ii. Device Management

Operating systems provide essential functions for managing devices connected to a computer. These functions include *allocating memory, processing input and output requests, and managing storage devices*. This device could be a keyboard, mouse, printer, or any other devices you may have connected.

An operating system will provide you with options to manage how each device behaves. For example, you can set up your keyboard to type in a specific language or make it so that the mouse only moves one screen at a time.

You can also use an operating system to install software and updates for your devices and manage their security settings.

iii. Process Management

The operating system's responsibility is to manage the processes running on your computer. This includes starting and stopping programs, allocating resources, and managing memory usage. The operating system ensures that the programs running on your computer should be compatible. It’s also responsible for enforcing program security, which helps to keep your computer safe from potential attacks.

How do Operating systems manage all processes?

Each process is given a certain amount of time to execute, called a ***quantum***. Once a process has used its quantum, the operating system interrupts it and provides another process with a turn. This ensures that each process gets a fair share of the CPU time.

The operating system manages processes by doing the following task:

- Allocating and deallocating the resources.
- Allocates resources such that the system doesn’t run out of resources.
- Offering mechanisms for process synchronization.

Helps in process communication (inter communication).

iv. Memory Management

One of the most critical functions of an operating system is **memory management**. This is the process of keeping track of all the different applications and processes running on your computer and all the data they’re using.

This is especially important on computers with a limited amount of memory, as it ensures that no application or process takes up too much space and slows down your computer. The operating system can move data around and delete files to make more space.

Operating systems perform the following tasks-

- Allocating/deallocating memory to store programs.
- Deciding the amount of memory that should be allocated to the program.
- Memory distribution while multiprocessing.
- Update the status in case memory is freed
- Keeps record of how much memory is used and how much is unused.

When a computer starts up, the operating system loads itself into memory and then manages all the other running programs. It checks how much memory is used and how much is available and makes sure that executing programs do not interfere with each other.

v. Job Management

An operating system’s (OS) job accounting feature is a powerful tool for tracking how your computer’s resources are being used. This information can help you pinpoint and troubleshoot any performance issues and identify unauthorized software installations.

Operating systems keep track of which users and processes use how many resources. This information can be used for various purposes, including keeping tabs on system usage, billing users for their use of resources, and providing information to system administrators about which users and processes are causing problems.

The operating system does the following tasks:

- Keeps record of all the activities taking place on the system.
- Keeps record of information regarding resources, memory, errors, resources, etc.
- Responsible for Program swapping(in and out) in memory
- Keeps track of memory usage and accordingly assigns memory
- Opening and closing and writing to peripheral devices.
- Creating a file system for organizing files and directories.

d. Objectives of an operating system.

i. Hardware Abstraction:

Objective: To provide a standardized interface between application programs and the hardware components (CPU, memory, disk drives, etc.), abstracting the complexity of hardware details for software developers.

ii. Resource Management:

Objective: Efficiently allocate and manage system resources, such as CPU time, memory space, input/output devices, and network bandwidth, to ensure optimal utilization and fair sharing among multiple processes or users.

iii. Process Management:

Objective: Create, schedule, and terminate processes, facilitating the execution of concurrent tasks. The OS manages the execution state, communication, and synchronization between processes.

iv. Memory Management:

Objective: Control and allocate the system's memory space to processes as needed. This includes managing virtual memory, swapping data between RAM and storage, and ensuring efficient memory utilization.

v. File System Management:

Objective: Provide an organized and structured way to store, retrieve, and manage files on storage devices. The OS handles file creation, deletion, reading, and writing, and maintains file integrity and security.

vi. Device Management:

Objective: Control and coordinate communication with input/output devices (e.g., printers, disk drives, network interfaces) by providing device drivers and managing device queues, ensuring efficient and secure data transfer.

vii. User Interface:

Objective: Offer a user-friendly interface for interaction with the computer system. This can be through a command-line interface (CLI) or a graphical user interface (GUI), simplifying the user experience.

viii. Security and Protection:

Objective: Enforce access controls, authentication, and authorization mechanisms to protect the system from unauthorized access and ensure data integrity. The OS also provides isolation between processes to prevent interference.

ix. Error Detection and Handling:

Objective: Detect, report, and handle errors or exceptional situations in the system, preventing crashes and providing mechanisms for recovery when possible.

x. System Performance Monitoring:

Objective: Monitor and analyze system performance, collecting data on resource usage, bottlenecks, and other metrics. This information helps optimize system performance and plan for future upgrades or changes.

xi. Communication and Networking:

Objective: Facilitate communication between different computers and devices in a networked environment, managing protocols and ensuring seamless data exchange.

xii. Support for Multi-User Environments:

Objective: Enable multiple users to interact with the system concurrently, providing mechanisms for user authentication, session management, and resource sharing.

e. Types of Operating System

i. Single User
   - Standard OS for a standalone computer system
   - only one person can use the computer system at any one time
   - can support multiple accounts but only one used at a time
   - processing power of CPU dedicated to the user
   - so multi-tasking can be performed i.e. running more than one program for that user at the same time

ii. Multi-User
   - More than one user accessing the system at the same time
   - most commonly: network operating system
   - server computer connected to a number of terminals
   - client-server LAN, ATM, ticket booking
   - O/S allocates a time slice to each user
   - switches from user to user
   - the larger the number of users the slower the system works
   - can run several programs simultaneously

iii. Multi-Tasking
   - CPU carries out more than one task at the same time, several programs can be loaded and running at the same time • e.g. using a word processor whilst browsing the Internet
   - Processor switches between tasks
     - completes part of one task then switches to do part of another task
     - this process happens very fast
     - appears that tasks are running concurrently
   - true multi-tasking requires more than one processor
     - many current CPUs have multiple processing cores

iv. Interactive
   - Direct user interaction whilst a program is running
     - accepts and responds to input from user
   - "question and answer" between user and computer
   - The computer gives the user an immediate response

v. Real-Time
   - Must respond to events with no discernible delay
   - Responds immediately to the data entered
   - Used where response time by the system is critical
     - where delays in processing would prove unacceptable
   - Used for control systems
     - traffic lights, air traffic control, aircraft navigation
   - Used for embedded systems
     - systems within another application,
     - e.g. cars and mobile phones

vi. Batch Processing

- Data can be collected over a period of time
- When all data collected -it is processed at the same time • as a ‘batch’
- Data is processed at an off-peak period
    - when there is less demand on the processor
    - less disruption to the daily work schedule
    - system & peripherals available when most needed
- Does not need staff intervention
- suitable for situations where there is a lot of similar data to be processed & the processing can wait:
    - cheque processing in a bank, gas bill printing, payroll processing •
- Not suitable where the data must be processed instantly data is not up to date until processing completed

vii. Distributed System

- Complex tasks are split into several smaller, similar tasks
- a series of computers connected together via network
    - computers can be in different physical locations
- each computer performs part of the processing
- results from individual machines are combined to achieve the overall task
- Examples:
    - film animation & CGI via “render farms”
    - Modeling real-world scenarios – scientific, environmental, and financial
    - analysis of large datasets

f. Handheld devices

Handheld operating systems are available in all handheld devices like Smartphones and tablets. It is sometimes also known as a Personal Digital Assistant. The popular handheld device in today’s world is Android and iOS. These operating systems need a high-processing processor and are also embedded with various types of sensors.

Some points related to Handheld operating systems are as follows:

i. Since the development of handheld computers in the 1990s, the demand for software to operate and run on these devices has increased.
ii. Three major competitors have emerged in the handheld PC world with three different operating systems for these handheld PCs.
iii. Out of the three companies, the first was the Palm Corporation with their PalmOS.
iv. Microsoft also released what was originally called Windows CE. Microsoft’s recently released operating system for the handheld PC comes under the name of Pocket PC.

v. More recently, some companies producing handheld PCs have also started offering a handheld version of the Linux operating system on their machines.

Features of Handheld Operating System:

i. Its work is to provide real-time operations.
ii. There is direct usage of interrupts.
iii. Input/Output device flexibility.
iv. Configurability.

Types of Handheld Operating System

i. Palm OS
ii. Symbian OS
iii. Linux OS
iv. Windows
v. Android

Palm OS:

Since the Palm Pilot was introduced in **1996**, the Palm OS platform has provided various mobile devices with **essential business tools**, as well as the capability that they can **access the internet via a wireless connection.**

These devices have mainly concentrated on providing basic personal-information-management applications. The latest Palm products have progressed a lot, packing in more storage, wireless internet, etc.

Symbian OS:

It has been the most widely-used smartphone operating system because of its ARM architecture before it was discontinued in **2014**. It was developed by Symbian Ltd.

This operating system consists of two subsystems where the first one is the **microkernel-based operating system** which has its associated libraries and the second one is **the interface of the operating system** with which a user can interact.

Since this operating system consumes much less power, it was developed for smartphones and handheld devices.

It has good connectivity as well as stability.

It can run applications that are written in Python, Ruby, .NET, etc.

Linux OS:

- Linux OS is an open-source operating system project which is a cross-platform system that was developed based on UNIX. It was developed by Linus Torvalds. It is a system software that allows the apps and users to perform some tasks on the PC.
- Linux is **free** and can be easily downloaded from the internet and it is considered that it has the best community support.
- Linux is **portable** which means it can be installed on different types of devices like mobile, computers, and tablets.
- It is a multi-user operating system.
- Linux **interpreter** program which is called **BASH** is used to execute commands.
- It provides user security using authentication features.

Windows OS:

Windows is an operating system developed by Microsoft. Its interface which is called Graphical User Interface eliminates the need to memorize commands for the command line by using a mouse to navigate through menus, dialog boxes, and buttons.

- It is named Windows because its programs are displayed in the form of a square. It has been designed for both a beginner as well professional.
- It comes preloaded with many tools that help the users to complete all types of tasks on their computer, mobiles, etc.
- It has a large user base so there is a much larger selection of available software programs.
- One great feature of Windows is that it is **backward compatible** which means that its old programs can run on newer versions as well.

Android OS:

It is a Google Linux-based operating system that is mainly designed for touchscreen devices such as phones, tablets, etc. There are three architectures which are **ARM**, **Intel**, and **MIPS** which are used by the hardware for supporting Android. These let users manipulate the devices intuitively, with movements of our fingers that mirror some common motions such as swiping, tapping, etc.

- Android operating system can be used by anyone because it is an open-source operating system and it is also free.
- It offers 2D and 3D graphics, GSM connectivity, etc.
- There is a huge list of applications for users since Play Store offers over one million apps.

Professionals who want to develop applications for the Android OS can download the Android Development Kit. By downloading it they can easily develop apps for Android.

Advantages of Handheld Operating System:

Some advantages of a Handheld Operating System are as follows:

- Less Cost.
- Less weight and size.
- Less heat generation.
- More reliability.

Disadvantages of Handheld Operating System:

Some disadvantages of Handheld Operating Systems are as follows:

- Less Speed.
- Small Size.
- Input / Output System (memory issue or less memory is available)

How Handheld operating systems are different from Desktop operating systems?

- Since the handheld operating systems are mainly designed to run on machines that have lower speed resources as well as less memory, they were designed in a way that they use less memory and require fewer resources.
- They are also designed to work with different types of hardware as compared to standard desktop operating systems. It happens because the power requirements for standard CPUs far exceed the power of handheld devices.
- Handheld devices aren't able to dissipate large amounts of heat generated by CPUs. To deal with such kind of problem, big companies like Intel and Motorola have designed smaller CPUs with lower power requirements and also lower heat generation. Many handheld devices fully depend on flash memory cards for their internal memory because large hard drives do not fit into handheld devices.

g. Design influence of security,

h. Networking

Networking refers to the practice of interacting with others to exchange information, develop contacts, and build relationships, especially to further one's career or business goals. Here are some key points to consider when it comes to networking:

1. Network Capabilities in Operating Systems:
   - Operating systems provide the foundational software layer that manages hardware resources and facilitates interactions between applications and the underlying hardware.
   - Most modern operating systems include built-in networking capabilities to enable communication between devices, data sharing, and resource access over networks.
   - Networking features in operating systems include support for Ethernet, Wi-Fi, Bluetooth, and other communication protocols necessary for connecting devices to networks.

2. Network Drivers and Protocols:
   - Operating systems contain network drivers that allow the hardware components (such as network interface cards) to communicate with the operating system's networking stack.
   - The networking stack within an operating system implements various network protocols and services required for transmitting data over networks. These protocols include TCP/IP, UDP, HTTP, FTP, DNS, and others.
   - The operating system's networking stack manages tasks such as packet routing, addressing, error detection, and data transmission/reception.

3. Network Configuration and Administration:
   - Operating systems provide tools and utilities for configuring and managing network settings, including IP addresses, DNS servers, gateways, and network interfaces.
   - Network administration tasks such as setting up firewalls, managing network security policies, and troubleshooting network connectivity issues often involve using built-in tools or third-party software integrated with the operating system.

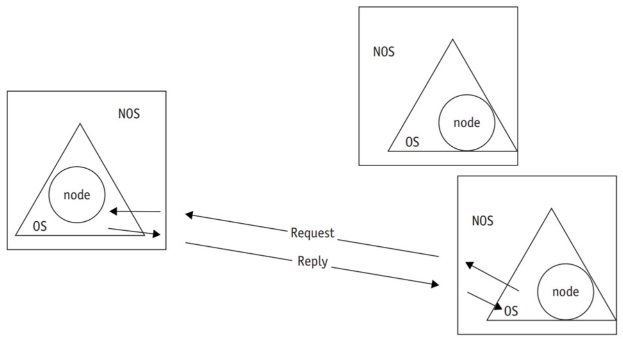
4. File and Resource Sharing:
   - Operating systems enable file and resource sharing over networks, allowing users to access files, printers, and other resources located on remote computers.
   - Network file systems (NFS) and server message block (SMB) protocols are commonly used for sharing files and directories between computers running different operating systems.
   - Operating systems implement authentication mechanisms and access control policies to regulate access to shared resources and ensure data security.

5. Integration with Network Services:
   - Operating systems interact with network services and applications running on local and remote computers. Examples of network services include web servers, email servers, domain controllers, and directory services.
   - Operating systems provide APIs (Application Programming Interfaces) and libraries that enable developers to create network-aware applications capable of communicating with other devices and services over networks.

Networks were created initially to share expensive hardware resources such as large mainframes, laser printers, and sizable hard disks. These physical networks, with their network operating systems, allowed organizations to increase the availability of these resources and spread the cost among many users. However, the focus of technology changed when system owners realized that a network's most prized resource wasn't the hardware—it was the information stored on it. Soon many operating systems were enhanced with network capabilities to give users throughout an organization easy access to centralized information resources. The network operating system was developed first, followed by the more powerful distributed operating system.

Network Operating System NOS

The network operating system (NOS) evolved from the need to give users global access to resources, globally manage the network's processes, and make the network almost completely transparent for users and their sites' operating systems, known as local operating systems. A typical NOS is shown in Figure below



It's important that the NOS accomplish this transparently. The local operating system views the action as having been performed onsite. That's because the network operating system handles the interfacing details and coordinates the remote processing. It also coordinates communications between the local operating systems by tracking the status and location of all entities in the system.

The local operating systems are traditional operating systems designed to run a single computer. That means they can perform a task only if the process is part of their environment; otherwise, they must pass the request on to the network operating system to run it. To a local operating system, it appears that the NOS is the server performing the task, whereas in reality the NOS is only the facilitator.

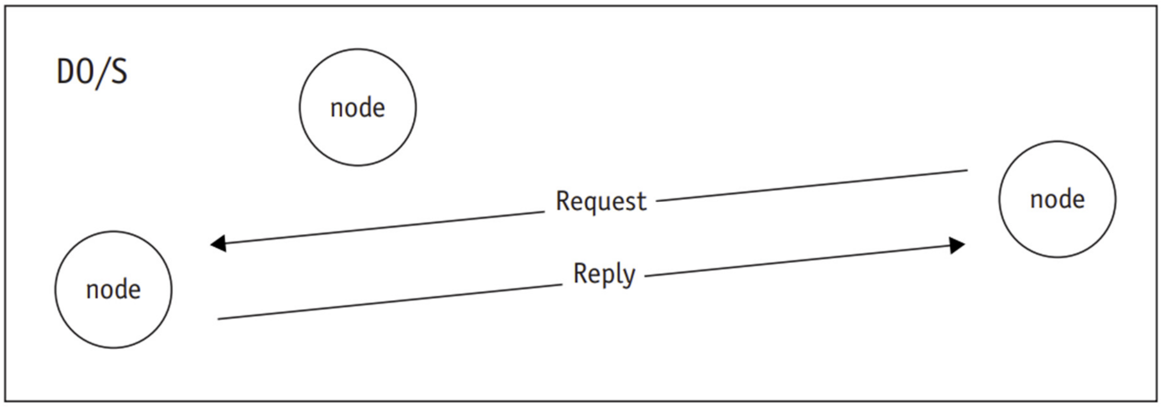
Disadvantages of NOS

- Does not take global control over the following. However, it sees them as autonomous local functions.
    - Memory management
    - Process Management
    - Device Management
    - File Management

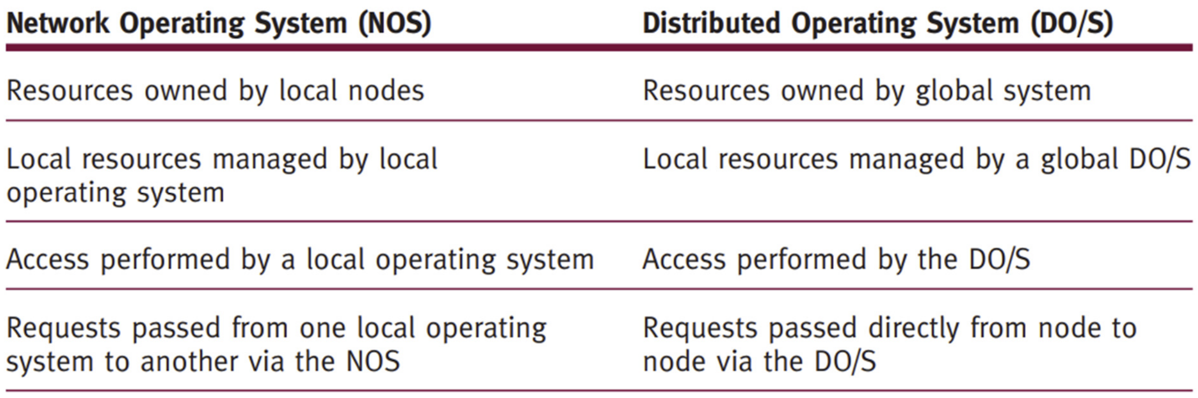
This disadvantage lead to creation of Distributed Operating System (DO/S)

Distributed Operating System (DO/S)

Distributed operating systems provide a unified environment designed to optimize operations for the network as a whole, not just for local sites, as illustrated in Figure xxx



The major difference between a NOS and a DO/S is how each views and manages the local and global resources. A NOS builds on capabilities provided by the local operating system and extends it to satisfy new needs. It accesses resources by using local mechanisms, and each system is controlled and managed locally based on that system's policy. On the other hand, the ***DO/S considers system resources to be globally owned and manages them as such***. It accesses resources using global mechanisms rather than local mechanisms, with system control and management based on a single system-wide policy. A comparison of the two types of systems is shown in Table 10.1.



Network Operating System (NOS)	Distributed Operating System (DO/S)
Resources owned by local nodes	Resources owned by global system
Local resources managed by local operating system	Local resources managed by a global DO/S
Access performed by a local operating system	Access performed by the DO/S
Requests passed from one local operating system to another via the NOS	Requests passed directly from node to node via the DO/S

i. Multimedia

Multimedia is the media that uses multiple forms of information content and information processing (eg. Text, audio, video , graphics, animation, interactivity) to inform or entertain the user. Multimedia also refers to the use of electronic media to store and experience multimedia content. Multimedia is similar to traditional mixed media in fine art, but with a broader scope.

Benefits of Multimedia

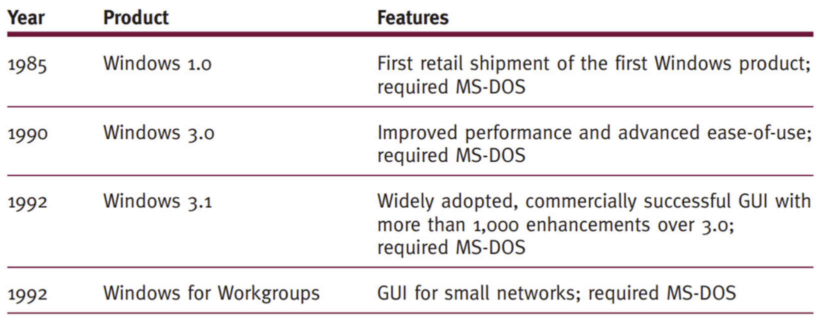
- Video Conferencing
- Distributed lecture
- Tele-medicine
- Cooperative work Environment
- Searching in large video and image database.

j. Windows

Windows is an operating system by Microsoft. It consist of GUI as a form of communication media to users

History of operating system

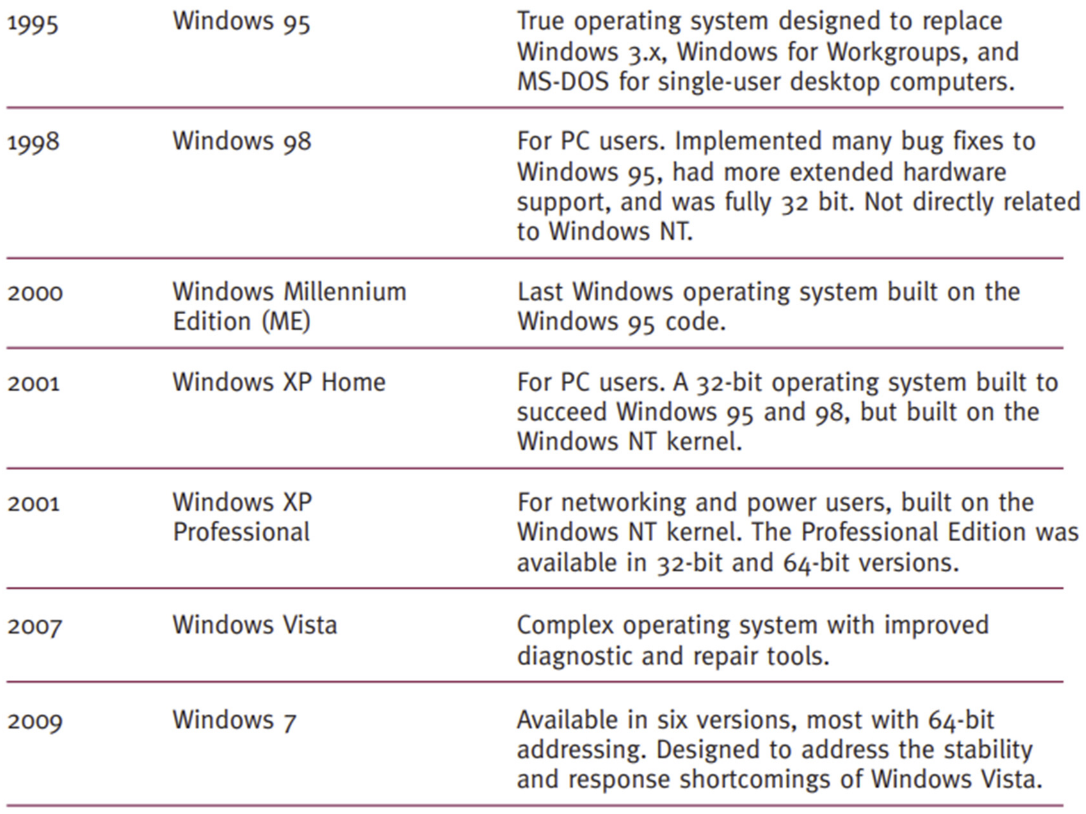
Windows 1.0 was introduced in 1985 and ran on microcomputers with the MS-DOS operating system. That is, the first Windows application was not a true operating system. It was merely an interface between the actual MS-DOS operating system and the user. Even though this was a simple product (when compared to the complex operating systems of today), it was notable because it was the first menu-driven interface for desktop computers that were compatible with the IBM personal computer (PC).



Year	Product	Features
1985	Windows 1.0	First retail shipment of the first Windows product; required MS-DOS
1990	Windows 3.0	Improved performance and advanced ease-of-use; required MS-DOS
1992	Windows 3.1	Widely adopted, commercially successful GUI with more than 1,000 enhancements over 3.0; required MS-DOS
1992	Windows for Workgroups	GUI for small networks; required MS-DOS

Before the release of the Windows 95 operating system, all Windows products were built to run on top of the MS-DOS operating system. That is, MS-DOS was the true operating system but took its direction from the Windows program being run on it. However, this layering technique proved to be a disadvantage. Although it helped Windows gain market share among MS-DOS users, MS-DOS had little built-in security, couldn’t perform multitasking, and had no interprocess communication capability. In addition, it was written to work closely with the microcomputer’s hardware, making it difficult to move the operating system to other platforms

To respond to these needs, Microsoft developed and released a succession of Windows operating systems (not mere GUIs) to appeal to home and office users, as shown in Table 15.2. (Parallel development of networking products is shown in Table 15.3.)



1995	Windows 95	True operating system designed to replace Windows 3.x, Windows for Workgroups, and MS-DOS for single-user desktop computers.
1998	Windows 98	For PC users. Implemented many bug fixes to Windows 95, had more extended hardware support, and was fully 32 bit. Not directly related to Windows NT.
2000	Windows Millennium Edition (ME)	Last Windows operating system built on the Windows 95 code.
2001	Windows XP Home	For PC users. A 32-bit operating system built to succeed Windows 95 and 98, but built on the Windows NT kernel.
2001	Windows XP Professional	For networking and power users, built on the Windows NT kernel. The Professional Edition was available in 32-bit and 64-bit versions.
2007	Windows Vista	Complex operating system with improved diagnostic and repair tools.
2009	Windows 7	Available in six versions, most with 64-bit addressing. Designed to address the stability and response shortcomings of Windows Vista.

In the fall of 1988, Microsoft hired David Cutler to lead the development of the Windows NT operating system. As an experienced architect of minicomputer systems, Cutler identified the primary market requirements for this new product: portability, multiprocessing capabilities, distributed computing support, compliance with government procurement requirements, and government security certification. The finished product has evolved as shown in Table above

In 1999, Microsoft changed the operating system's name from Windows NT to Windows 2000, which was available in four packages: Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, and Windows 2000 Datacenter Server. The Datacenter Server was a new product designed for large data warehouses and other data-intensive business applications, and supported up to 64GB of physical memory. Likewise, Windows Server 2003 was also released with these same four packages plus a Web edition. Windows Server 2008 Release 2 was launched in 2009 to coincide with the launch of Windows 7 and offered improved support for multiple cores, up to 64, reduced power consumption, and increased virtualization capabilities

k. Kernel

i. What is Kernel?

A Kernel can be defined as an intermediary between applications and hardware. This means that applications can run without knowing or caring about the underlying hardware details. It manages low-level tasks such as disk management, task management, and **memory management**. Whenever you start a system, Kernel is the first program that is loaded after the bootloader, and it remains in the memory until the **operating system** is shutdown.

The Kernel is seen as the most important part of your operating system. It's responsible for managing your system resources, running processes, and communicating with the hardware. The Kernel interacts with devices, allocating memory and handling interrupts and requests from processes running on the computer. The Kernel manages resources such as *CPU time, disk space, network connections, tim***ers,** *etc.*, executes user-level programs at regular intervals (via scheduler), recognizes files when users open them, and intercepts certain keystrokes before they are sent to the application program in focus, etc. The core component keeps your system running, managing all your software and hardware. Without it, your computer would be a useless pile of metal and plastic.

ii. Functions of a Kernel

A Kernel in an operating system performs the following functions:

- **Device Management:** Processes require various peripheral devices such as a mouse and keyboard connected to the computer to perform various tasks. The Kernel manages the allocation of the peripheral devices.
- **Resource Management:** Kernel shares the resources between different processes while ensuring that every process has uniform access to the resources
- **Memory management:** Every process requires some memory to execute. The Kernel allows the processes to access the memory safely. It is responsible for allocating and deallocating memory to processes for their execution.
- **Access Computer Resource:** A kernel can access different computer resources such as RAM, CPU, I/O devices, and other resources. Every program needs to access the memory to execute. The Kernel decides which

memory each process will use, and the action is taken if memory is unavailable.

iii. Advantages of the Kernel
   - It is responsible for managing the system's resources, including memory, processors, and devices.
   - Provides essential services to users and applications, such as security, multitasking, and communications.
   - It handles essential tasks such as suspending or terminating processes when needed.
   - It improves performance, better security, and increased stability.

iv. Disadvantages of the Kernel
   - The developer should need to consider the particular requirements of the device. But, not all developers are familiar with the specific hardware requirements of each device.
   - The Kernel can be vulnerable to attacks. If an attacker can exploit a vulnerability in the Kernel, they could gain access to your system and damage or steal your data.

v. User and Kernel Mode
   - User Mode

   When an application like MS word handling a text editor runs on the operating system, the system is in user mode. The mode bit is 0 in that case. The mode bit will change from 1 to 0 if it switches to kernel mode. The user mode to kernel mode is switched when there is a need to access any device or any interrupt/system call occurs.

   - Kernel-mode

   When the system boots, the system starts with kernel mode. And the applications are executed in user mode. The switch can happen from user mode to kernel mode when

     - *Interrupt occurs*
     - *Need to access privileged instruction*
     - *Need to access any device*
     - *The mode bit is changed from 0 to 1 in case of the system switch back to user mode.*

vi. Types of the Kernel

The five main types of kernels are:

   - Monolithic Kernel

- Microkernel
- Hybrid Kernel
- Nano Kernel
- Exo Kernel

Monolithic Kernel: Monolithic kernels are the simplest and most common type of Kernel. They include the core functionality of the operating system and support all devices connected to it. In this, the user and kernel services are implemented in the same memory space. Due to this, the size of the kernel increases, which in turn increases the size of the operating system. The main benefit is that the process execution is faster as there is no separate memory space for the user and Kernel.

Microkernel: Microkernels are a newer development and, as such, are not as common as monolithic kernels. They include only the essential services and devices required for the system to function. This results in a smaller kernel that is faster and uses less memory. Here, the user and kernel services are implemented in two different spaces. It has separate User Space and Kernel Space. This reduces the size of the Kernel and results in reducing the size of the operating system.

Hybrid Kernel: Hybrid kernels combine monolithic and microkernels. They include more services than microkernels but less than monolithic kernels. This allows them to offer some of the benefits of both kernels. It borrows speed from the monolithic kernels and modularity from microkernels.

Nano Kernel: Nano kernels are the smallest type of Kernel, consisting of only a few thousand lines of code. It means that the code executing in the privileged mode of the hardware is minimal. They are used primarily in embedded systems or devices with limited resources.

ii. Process Management:

a. Concept of a process,

What is a Process?

A process is a program in the execution phase. A process is created, executed, and terminated. A Process goes through these states. We will discuss these states with the explanation. So, in the short process keeps on changing the states. According to a process’s activity, the process’s state is determined.

The process is categorized into two

1) System Processes

System processes are started by the operating system, and they usually have something to do with running the OS itself.

2) User Processes

User processes, as you might guess, are those that the user starts. The operating system though handles various system processes, but it mainly handles the user code execution.

Each process can run independently of other processes and have its own memory space. This makes process management tricky, as the OS needs to ensure that each process gets its fair share of resources (memory, CPU time, etc.) while ensuring that the system remains stable.

b. Process Management

It is an important part of the operating system. It allows you to control the way your computer runs by managing the currently active processes. This includes ending processes that are no longer needed, setting process priorities, and more. You can do it on your computer also.

There are a few ways to manage your processes. The first is through the use of Task Manager. This allows you to see all of the processes currently running on your computer and their current status and CPU/memory usage. You can end any process that you no longer need, set a process priority, or start or stop a service.

c. Scheduling, and synchronization.

Operating system scheduling is a critical aspect of the system's functionality, responsible for efficiently managing and allocating resources to various tasks or processes competing for the CPU's attention. Scheduling is essential for optimizing system performance, responsiveness, and fairness. Here are some key concepts related to operating system scheduling:

a) Process Scheduling:

Process scheduling involves selecting the next process to execute on the CPU. The operating system maintains a queue of processes and uses scheduling algorithms to determine the order in which they get CPU time.

b) Scheduling Algorithms:

Various scheduling algorithms are employed to determine the order in which processes are dispatched. Common algorithms include:

First-Come-First-Serve (FCFS): Processes are executed in the order they arrive.

Shortest Job Next (SJN) or Shortest Job First (SJF): The process with the shortest burst time is executed next.

Round Robin (RR): Each process gets a small unit of CPU time, and then the next process in the queue is given a turn.

Priority Scheduling: Processes are assigned priorities, and the one with the highest priority is selected for execution.

Multilevel Queue Scheduling: Processes are divided into priority levels, and each queue has its scheduling algorithm.

c) Preemptive vs. Non-preemptive Scheduling:

In preemptive scheduling, the operating system can interrupt a currently running process to start or resume another, usually due to a higher-priority task becoming available. In non-preemptive scheduling, a process runs until it completes or voluntarily releases the CPU.

d) Context Switching:

When the CPU switches from one process to another, a context switch occurs. This involves saving the state of the currently running process and loading the state of the new process. Context switches introduce overhead, so minimizing them is crucial for efficiency.

e) CPU Burst:

The time a process spends executing before it needs to access I/O or wait for an event is called a CPU burst. Scheduling algorithms often aim to minimize the turnaround time by optimizing for short CPU bursts.

f) Starvation:

Starvation occurs when a process is repeatedly postponed in favor of others, leading to the process not getting sufficient CPU time. Scheduling algorithms need to address starvation to ensure fairness.

g) Fair Scheduling:

Fair scheduling aims to allocate CPU time fairly among competing processes, preventing any single process from monopolizing resources. This is crucial for ensuring a responsive and equitable system.

h) Priority Inversion:

Priority inversion is a situation where a low-priority task holds a resource needed by a high-priority task. This can lead to unexpected delays for the high-priority task.

i) Real-time Scheduling:

Real-time scheduling is designed to meet deadlines for time-sensitive tasks. Hard real-time systems have strict deadlines, while soft real-time systems have less stringent requirements.

j) Multicore Scheduling:

Modern systems often have multiple CPU cores. Operating systems need to efficiently distribute processes across these cores for parallel execution, optimizing overall system performance.

Efficient scheduling is crucial for maintaining system responsiveness, meeting performance goals, and ensuring fair resource allocation in an operating system. The choice of scheduling algorithm depends on the specific requirements and characteristics of the system and the applications it supports.

i. Functionality Mechanisms to support client-server models

Client-server models in operating systems involve a distributed computing architecture where tasks and responsibilities are divided between client devices and server systems. The operating system plays a crucial role in supporting and facilitating the functionality of client-server models. Here are some mechanisms and functionalities that operating systems employ to support client-server architectures:

a) Process Synchronization:

Operating systems manage concurrent processes on both client and server sides. Synchronization mechanisms, such as locks, semaphores, and mutexes, help coordinate access to shared resources to avoid conflicts and ensure consistency.

Locks:

Locks is a synchronization primitive that prevents the state from being modified or accessed by multiple threads of execution at once.

In computer science, a lock or mutex (from mutual exclusion) is a synchronization primitive that prevents the state from being modified or accessed by multiple threads of execution at once. Locks enforce mutual exclusion concurrency control policies, and with a variety of possible methods there exists multiple unique implementations for different applications.

Semaphores:

Semaphores refer to the integer variables that are primarily used to solve the critical section problem via combining two of the atomic procedures, wait and signal, for the process synchronization

Semaphores are just normal variables used to coordinate the activities of multiple processes in a computer system. They are used to enforce mutual exclusion, avoid race conditions, and implement synchronization between processes.

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.

Semaphores are used to implement critical sections, which are regions of code that must be executed by only one process at a time. By using semaphores, processes can coordinate access to shared resources, such as shared memory or I/O devices.

A semaphore is a special kind of synchronization data that can be used only through specific synchronization primitives. When a process performs a wait operation on a semaphore, the operation checks whether the value of the semaphore is >0. If so, it decrements the value of the semaphore and lets the process continue its execution; otherwise, it blocks the process on the semaphore. A signal operation on a semaphore activates a process blocked on the semaphore if any, or increments the value of the semaphore by 1. Due to these semantics, semaphores are also called counting semaphores. The initial value of a semaphore determines how many processes can get past the wait operation.

b) Inter-process Communication (IPC):

IPC mechanisms enable communication between processes on the client and server. This includes message passing, shared memory, and sockets for network communication. The OS provides APIs and services for IPC to facilitate data exchange between client and server applications.

c) Network Protocols:

Operating systems include network protocol stacks that support communication between clients and servers over a network. Common protocols include TCP/IP for reliable communication and UDP for more lightweight, connectionless communication.

d) Socket Programming:

Socket APIs allow applications to create, connect, and communicate over network sockets. This enables client-server communication through established network protocols. The OS provides the necessary infrastructure for socket programming.

e) File and Resource Sharing:

The operating system facilitates file and resource sharing between clients and servers. Network file systems, such as NFS (Network File System) or SMB (Server Message Block), allow clients to access files and resources on remote servers transparently.

f) Security Mechanisms:

Operating systems implement security features to protect data and ensure the integrity of communication in client-server models. This includes user authentication, encryption, and access control mechanisms to prevent unauthorized access and data breaches.

g) Concurrency Control:

Concurrency control mechanisms manage multiple client requests and server responses concurrently. The operating system ensures proper scheduling and allocation of resources to handle multiple client-server interactions efficiently.

h) Fault Tolerance and Reliability:

Operating systems may include features to enhance fault tolerance and reliability in client-server interactions. This can involve mechanisms like transaction management, data replication, and recovery procedures to handle failures gracefully.

i) Load Balancing:

In server clusters or distributed systems, the OS may incorporate load balancing mechanisms to distribute client requests across multiple servers evenly. This ensures optimal resource utilization and improves system performance.

j) Remote Procedure Call (RPC):

RPC mechanisms enable processes to execute code on a remote server as if they were local. The OS provides support for RPC, allowing client applications to invoke procedures or functions on server systems seamlessly.

k) Distributed File Systems:

Distributed file systems extend file-sharing capabilities across multiple servers and clients. The OS manages access to distributed file systems and ensures consistency in data storage and retrieval