Unit 4: Computer Software

Introduction

Computer software refers to a set of instructions and programs that tell a computer what to do. Software is a critical component of modern computing, enabling users to perform a wide range of tasks, from browsing the web to creating complex scientific simulations.

There are two primary types of software: system software and application software. System software includes the operating system (OS) and utility programs that manage computer hardware resources and provide a platform for other software to run on. Application software, on the other hand, refers to programs that perform specific tasks for users, such as word processors, spreadsheets, and web browsers.

Software is typically created by software developers using programming languages such as Java, Python, C++, and many others. Once the software is created, it is typically distributed to users through various means, including physical media like CDs and DVDs, downloads from the internet, or through a subscription service.

Software is constantly evolving, with new features and capabilities being added regularly. Keeping software up-to-date is important to ensure that it remains secure and functional, and to take advantage of new features and enhancements that are added over time.

Types of Software

There are two primary types of software: system software and application software.

1. System software: This type of software is responsible for managing computer hardware resources and providing a platform for other software to run on. Some examples of system software include:

- Operating systems: such as Windows, macOS, Linux, and iOS
- Device drivers: software that allows computer hardware to communicate with the operating system and other software
- Firmware: software that is embedded in hardware devices and provides low-level functionality, such as booting up the device
- Utility programs: software that performs system-level tasks such as disk formatting, backup and recovery, and security management

- **Developing Software:** Developing software involves a process of designing, coding, testing, and maintaining software applications. The process can be divided into several stages, each with its own set of activities and deliverables
    A. **Programming Language:** A programming language is a formal language used to communicate instructions to a computer. Programming languages are used to create software applications, websites, and other computer programs. There are many programming languages, each with its own syntax, grammar, and rules for writing code.

    B. **Language Translator:** A language translator is a program or software that translates code written in one programming language into code written in another programming language. There are three main types of language translators:
        - Compiler: A compiler is a language translator that takes source code written in a high-level programming language and translates it into machine code, which can be directly executed by the computer. The compiled code can then be run without the need for the original source code or the compiler.
        - Interpreter: An interpreter is a language translator that reads the source code of a program and executes it directly, without compiling it into machine code first. The interpreter translates the code line by line, and executes it as it goes along. Interpreted code tends to be slower than compiled code, but it can be more flexible and easier to debug.
        - Assembler: An assembler is a program or software that translates assembly language code into machine code that can be executed by the computer's processor. Assembly language is a low-level programming language that is used to write programs that interact with the computer's hardware directly, such as device drivers and operating systems.

        An assembler takes the assembly code, which is written using mnemonic codes that represent machine language instructions, and translates them into binary machine code that can be executed by the processor. This process is called assembly or assembling.

C. Linker

A linker is a program or software that combines object files generated by a compiler into a single executable file that can be run on a computer. An object file is a compiled code file that contains machine code and data, but is not yet executable.

The linker performs several important tasks, including resolving external references, merging multiple object files, and generating the final executable file.

When a program is compiled, it may reference functions or variables that are defined in other files or libraries. These external references are not resolved until the linking stage, where the linker resolves them by locating the corresponding functions or variables and linking them to the program.

The linker also merges multiple object files into a single executable file. This allows different parts of the program to be compiled separately and then combined into a single file for easier distribution.

Finally, the linker generates the final executable file by linking the object files together and adding any necessary information, such as the program's entry point and any libraries that the program requires.

Linkers are an essential part of the software development process, as they enable developers to create executable files that can be run on a computer. Without linkers, it would be necessary to manually link object files together, which would be a time-consuming and error-prone process.

D: Loader

A loader is a program or software that loads an executable file into memory and prepares it for execution. When an executable file is run on a computer, the operating system uses a loader to perform several important tasks:

1. Memory allocation: The loader allocates memory for the program in the computer's RAM.
2. Relocation: The loader adjusts the program's memory references so that they point to the correct memory locations.
3. Symbol resolution: The loader resolves any symbols used by the program that were not resolved during the linking stage.
4. Dynamic linking: The loader may dynamically link any shared libraries that the program requires.
5. Initialization: The loader initializes the program's data and sets up any necessary runtime environment.

Once the loader has completed these tasks, the program is ready to be executed by the processor. The loader passes control of the program to the operating system, which starts executing the program's instructions.

Loaders are an essential part of the software development process, as they enable developers to run their programs on a computer. Without loaders, it would be necessary to manually load programs into memory and set up the necessary runtime environment, which would be a time-consuming and error-prone process.

2. **Application software:** This type of software is designed to perform specific tasks for users. There are many different types of application software, including:

- Productivity software: programs such as word processors, spreadsheets, and presentation software that are used for creating documents, managing data, and making presentations
- Graphics software: programs such as photo editors, 3D modeling software, and video editing software that are used for creating and editing images and videos

- Communication software: programs such as email clients, messaging apps, and video conferencing software that are used for communicating with others
- Entertainment software: programs such as video games and multimedia software that are used for entertainment purposes
- Education software: programs such as e-learning platforms and educational games that are used for teaching and learning

Overall, there are many different types of software available, each with its own unique set of features and capabilities.

Software Acquisition

Software acquisition refers to the process of obtaining or acquiring software for use in a business or personal setting. There are several ways to acquire software, including:

1. Commercial off-the-shelf (COTS) software: This is pre-built software that is available for purchase from software vendors. COTS software is designed to be used by a wide range of users and is often sold with user licenses that specify how many users can use the software.
2. Open-source software: This is software that is made available to the public with its source code, allowing users to modify and distribute the software as they see fit. Open-source software is often free to use, although some open-source projects may require donations or contributions.
3. Custom software development: This involves hiring a software developer or development team to create software specifically tailored to the needs of a business or organization.
4. Cloud-based software: This refers to software that is hosted and accessed over the internet rather than installed on a local computer. Cloud-based software is often sold on a subscription basis and can be accessed from anywhere with an internet connection.

When acquiring software, it's important to consider factors such as the cost, licensing requirements, support and maintenance, and compatibility with existing systems. It's also important to ensure that the software meets the specific needs and requirements of the organization or individual, and that it is secure and reliable.

Programming Languages

A programming language is a set of instructions that are used to create computer programs, scripts, and software. There are many programming languages available, each with its own syntax, features, and use cases. Some of the most popular programming languages include:

1. Java: This is a general-purpose, object-oriented programming language that is designed to be platform-independent. Java is commonly used for building web applications, mobile apps, and enterprise software.
2. Python: This is a high-level, interpreted programming language that is known for its simplicity and readability. Python is commonly used for data analysis, artificial intelligence, scientific computing, and web development.
3. C++: This is a general-purpose, object-oriented programming language that is commonly used for developing system software, games, and other high-performance applications. C++ is known for its speed and efficiency.
4. JavaScript: This is a high-level, interpreted programming language that is commonly used for building web applications and interactive user interfaces. JavaScript is often used alongside HTML and CSS to create dynamic and interactive web pages.
5. C#: This is a modern, object-oriented programming language that is designed to be easy to learn and use. C# is commonly used for building Windows applications, video games, and mobile apps.
6. PHP: This is a server-side scripting language that is commonly used for building dynamic web applications and websites. PHP is often used alongside HTML and CSS to create dynamic and interactive web pages.
7. Swift: This is a modern, high-level programming language that is designed specifically for building iOS and macOS applications. Swift is known for its simplicity and ease of use.

Overall, there are many programming languages available, each with its own strengths and weaknesses. The choice of programming language depends on the specific requirements of the project, the skill level of the developer, and the available resources.

Operating System

Introduction

An operating system (OS) is a software program that manages computer hardware resources and provides common services for computer programs. The primary function of an operating system is to act as a bridge between computer hardware and software, allowing applications to interact with the hardware without needing to know the specifics of the underlying hardware.

Some of the main functions of an operating system include:

1. Memory management: The operating system manages the allocation and deallocation of memory for applications and system processes.
2. Processor management: The operating system schedules tasks and manages the use of the processor by different programs and processes.
3. Input/output management: The operating system manages input and output operations between hardware devices and software applications.
4. File management: The operating system manages the storage and retrieval of files on storage devices such as hard drives and solid-state drives.
5. Security management: The operating system provides security features such as user authentication and access control to ensure the security and privacy of data on the system.

Examples of popular operating systems include:

1. Windows: This is a series of operating systems developed by Microsoft and used on a wide range of computers, including desktops, laptops, and servers.
2. macOS: This is the operating system used on Apple's Macintosh computers.
3. Linux: This is a family of open-source operating systems that are widely used on servers, supercomputers, and mobile devices.
4. Android: This is a mobile operating system developed by Google and used on a wide range of smartphones and tablets.

Objectives of Operating System

The main objectives of an operating system (OS) are to manage computer hardware resources, provide a user-friendly interface for running applications, and ensure the security and stability of the system. Some of the specific objectives of an OS include:

1. Resource management: The operating system manages computer hardware resources such as the processor, memory, and storage devices to ensure that each application gets the resources it needs to run efficiently. The OS also manages input/output operations between devices and applications.
2. Memory management: The operating system manages the allocation and deallocation of memory for applications and system processes. It ensures that each application has access to the memory it needs, and that memory is not wasted or used inefficiently.
3. Processor management: The operating system schedules tasks and manages the use of the processor by different programs and processes. It ensures that each application gets enough processing time to run smoothly and efficiently.
4. File management: The operating system manages the storage and retrieval of files on storage devices such as hard drives and solid-state drives. It provides a file system that allows users and applications to access and manipulate files.
5. Security management: The operating system provides security features such as user authentication and access control to ensure the security and privacy of data on the system. It also protects the system from malicious software and unauthorized access.
6. User interface: The operating system provides a user-friendly interface for running applications and interacting with the computer. It allows users to launch applications, access files, and perform system tasks using a graphical user interface (GUI) or a command-line interface (CLI).

Types of Operating System

There are several types of operating systems (OS) available, each with its own strengths and weaknesses. The main types of operating systems are:

1. Batch Operating Systems: Batch operating systems are designed to process large volumes of data in batches. They are commonly used in applications such as payroll processing, inventory management, and billing. In a batch operating system, jobs are submitted in batches, and the operating system executes them without user interaction. Batch operating systems are designed to maximize the utilization of resources such as the CPU and storage devices. Examples of batch operating systems include IBM's OS/360 and UNIVAC's EXEC 8.

2. Real-time Operating Systems: Real-time operating systems are designed to handle real-time applications that require fast and predictable response times. Real-time operating systems are used in applications such as industrial control systems, telecommunications, and military systems. In a real-time operating system, tasks are executed as soon as they are received, with a guaranteed response time. Real-time operating systems are optimized for fast response times and predictable behavior, and they are designed to minimize the latency between when an event occurs and when the system responds to it. Examples of real-time operating systems include VxWorks, QNX, and RTLinux.

3. **Distributed Operating System:** A distributed operating system (DOS) is an operating system that manages a network of independent computers and makes them appear to the user as a single system. In a DOS, each computer in the network has its own local operating system, but the computers work together to provide a unified computing environment.

   The primary goal of a distributed operating system is to provide transparency to the users and applications. This means that users and applications can access resources, such as files and printers, on any computer in the network without needing to know which computer the resource is located on. The distributed operating system manages the network resources and makes them available to users and applications as if they were local resources.

   A distributed operating system typically provides the following features:

   1. Resource sharing: The ability to share hardware and software resources across the network, such as storage devices, printers, and processing power.
   2. Distributed file systems: The ability to access files on remote computers as if they were local files.
   3. Process migration: The ability to move processes between computers in the network, in order to balance workload and improve performance.
   4. Fault tolerance: The ability to recover from hardware and software failures by automatically redirecting requests to alternative resources.
   5. Security: The ability to ensure that users and applications can only access the resources they are authorized to access.

Distributed operating systems are used in large-scale computing environments, such as data centers and cloud computing platforms, where the use of multiple computers is necessary to provide the required processing power and storage capacity.

4. **Multitasking OS:**

A multitasking operating system (OS) is an operating system that allows multiple programs or processes to run concurrently on a computer. A multitasking OS is designed to maximize the use of a computer's resources by efficiently sharing its processing power, memory, and input/output devices among multiple programs.

In a multitasking OS, the computer's CPU time is divided into multiple time slices, and each program or process is given a portion of CPU time to execute. The OS manages the allocation of CPU time and ensures that each program or process receives a fair share of CPU time.

Multitasking operating systems can be further classified into two categories:

1. Preemptive multitasking: In preemptive multitasking, the operating system controls the allocation of CPU time and can interrupt a running program or process if a higher-priority task needs to run. This ensures that critical tasks are executed on time, even if other less important tasks are running.
2. Cooperative multitasking: In cooperative multitasking, the programs or processes voluntarily yield control of the CPU to other programs or processes. The operating system does not interrupt a program or process unless it explicitly yields control or

becomes unresponsive. This approach can be more efficient, but it also requires that all programs or processes cooperate and yield control appropriately.

Multitasking operating systems allow users to run multiple applications simultaneously, improving productivity and efficiency. They are widely used in desktop and mobile computing, as well as in servers and other large-scale computing environments.

5. **Network OS**

A network operating system (NOS) is an operating system that is designed to manage and control network resources. A NOS allows multiple computers to communicate and share resources such as printers, files, and applications, by providing the necessary protocols and services to support network operations.

Some of the features of a network operating system include:

1. Network services: A NOS provides a variety of network services such as file sharing, printing, email, and directory services.
2. Security: A NOS provides security features such as authentication, authorization, and access control to ensure that only authorized users can access network resources.
3. Network management: A NOS provides tools to manage and monitor the network, such as network performance monitoring, network diagnostics, and network backup and recovery.
4. Distributed processing: A NOS provides the ability to distribute processing tasks across multiple computers, improving performance and scalability.
5. Resource sharing: A NOS allows network resources such as printers, files, and applications to be shared among multiple users and computers.

Examples of network operating systems include Microsoft Windows Server, Linux, Novell NetWare, and IBM AIX. Network operating systems are used in a wide range of environments, including small businesses, large enterprises, and government organizations.

Functions of OS

An operating system (OS) performs a wide range of functions, some of which include:

1. Memory management: The OS manages the computer's memory, allocating and deallocating memory as required by the programs and processes running on the system.

This includes managing virtual memory, which allows programs to access more memory than is physically available.
2. Processor management: The OS manages the computer's processors, scheduling processes and threads to run on the available processors. The OS also manages process synchronization, which prevents two processes from accessing the same resource at the same time.
3. Input/output management: The OS manages input and output operations, providing drivers and interfaces for devices such as keyboards, mice, displays, and storage devices. It also manages network communications, allowing the computer to communicate with other computers on the network.
4. File management: The OS manages the computer's file system, including creating, deleting, and renaming files and directories. The OS also provides access controls and security features to ensure that only authorized users can access files and directories.
5. User interface: The OS provides a user interface (UI) that allows users to interact with the computer, using graphical or command-line interfaces. The UI includes features such as windows, menus, and dialog boxes, and may include support for speech and touch input.
6. System services: The OS provides a range of system services, including printing, backup and restore, and system updates. These services are often provided through the use of utility programs or system tools.

Overall, the OS provides a layer of abstraction between the hardware of the computer and the applications and users that interact with it. It manages the computer's resources and provides a stable and secure environment for applications to run.

Examples of Operating System (OS)

There are many examples of operating systems (OS) that are used on various devices and systems. Some of the most commonly used operating systems include:

1. Microsoft Windows: A popular operating system for personal computers, laptops, and servers. Different versions of Windows include Windows 10, Windows 8, Windows 7, and Windows Server.
2. macOS: An operating system used exclusively on Apple's Macintosh computers, including desktop and laptop models.
3. Linux: A free and open-source operating system used on a wide range of devices, including servers, desktops, and embedded systems.
4. Android: A mobile operating system based on the Linux kernel, used on smartphones, tablets, and other mobile devices.
5. iOS: An operating system used exclusively on Apple's mobile devices, including the iPhone and iPad.

6. Unix: An operating system originally developed in the 1960s, Unix is still used today on servers and workstations in many organizations.
7. Chrome OS: A lightweight operating system designed by Google for use on Chromebook laptops.
8. IBM z/OS: An operating system designed for IBM mainframe computers used in large organizations for mission-critical applications.

These are just a few examples of the many operating systems that are in use today. Each operating system has its own unique features and advantages, and is chosen based on the needs of the user or organization.

New Trends in Software

There are several new trends in software development that are shaping the way software is created and used. Here are some examples:

1. Artificial intelligence (AI): AI is becoming increasingly integrated into software development, with developers using AI to automate tasks such as testing, debugging, and performance optimization.
2. Low-code and no-code development: Low-code and no-code development platforms are making it easier for non-technical users to create software applications, allowing businesses to rapidly develop and deploy applications without the need for specialized development skills.
3. DevOps: DevOps is a software development methodology that emphasizes collaboration and communication between developers and operations teams, with the goal of increasing the speed and efficiency of software development and deployment.
4. Cloud computing: Cloud computing has revolutionized the way software is developed, deployed, and used. Cloud-based development platforms and infrastructure services have made it easier and more cost-effective for businesses to develop, test, and deploy software.
5. Blockchain: Blockchain technology is being used to create decentralized applications (dApps) that can be used to build secure and transparent systems for a wide range of applications, including financial transactions, supply chain management, and identity management.
6. Internet of Things (IoT): IoT is a rapidly growing field that involves the integration of sensors, devices, and other technologies to create intelligent systems that can interact with the physical world. Software developers are playing a key role in creating the applications and systems that make IoT possible.