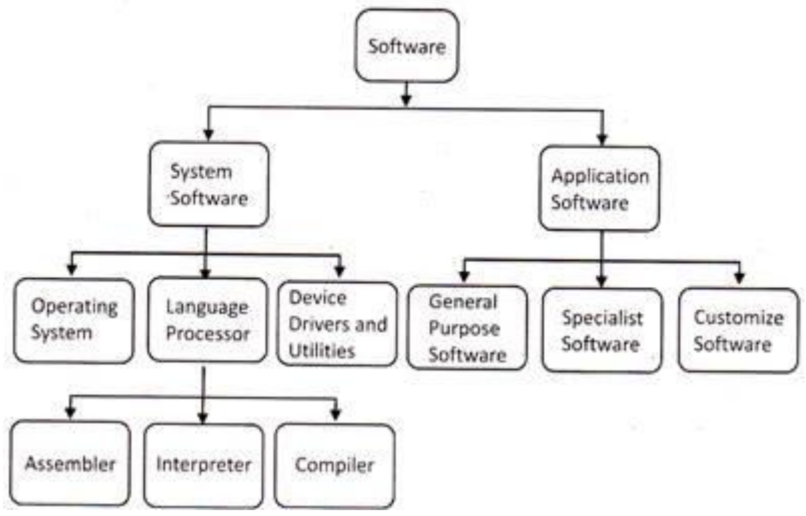
Table of Contents

Software



System software

System software is a type of computer program that is designed to run a computer's hardware and application programs. If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications. The operating system is the best-known example of system software. The OS manages all the other programs in a computer [1].

Language translator

Mostly, high-level languages like Java, C++, Python, and more are used to write the programs, called source code, as it is very uninteresting work to write a computer program directly in machine code. These source codes need to translate into machine language to be executed because they cannot be executed directly by the computer [2].

Types of language translator.

Compiler

A compiler is a specialized software tool that translates high-level source code written in a programming language into machine code or an intermediate code that can be executed by a computer [3].

Interpreter

An interpreter is a software program that directly executes the source code of a high-level programming language without the need for a separate compilation step. Instead of translating entire source code into machine code before execution, an interpreter processes the code line-by-line or statement-by-statement, interpreting and executing each part on-the-fly [3].

Assembler

An assembler is a software tool that translates assembly language programs into machine code or executable code that can be run on a computer's central processing unit (CPU). Assembly language is a low-level programming language that is closely tied to the architecture of a specific CPU. Each assembly language instruction corresponds to a specific machine language instruction [3].

Generation of programming languages

The generation of programming languages can be categorized into multiple generations, each representing a stage in the evolution of programming languages. Here's a brief overview:

First Generation (1940s-1950s):

Machine Language: The earliest programming languages were machine languages, directly represented as binary code that the computer's central processing unit (CPU) could execute. Programming involved entering sequences of 0s and 1s [3].

Second Generation (1950s-1960s):

Assembly Language: Assembly languages were introduced to make programming more human-readable. Instead of binary code, symbolic mnemonics and symbols represented instructions, making it easier for programmers to work with the hardware [3].

Third Generation (1960s-1980s):

High-Level Languages: High-level programming languages, such as Fortran, COBOL, and ALGOL, were developed. These languages allowed programmers to write code using more abstract, English-like syntax. Compilers were used to translate high-level code into machine code [3].

Fourth Generation (1980s-Present):

Declarative and Scripting Languages: Fourth-generation languages focused on higher levels of abstraction, allowing developers to specify what they want to achieve without detailing how. Database query languages, such as SQL, and scripting languages, like Python and Ruby, fall into this category [3].

Fifth Generation (Present and Future):

AI and Logic Programming: The fifth generation is characterized by languages designed to support artificial intelligence and logic programming. Prolog, a logic programming language, is an example. Additionally, languages used for parallel and distributed computing, as well as those designed for specific domains like scientific computing or web development, also fall into this generation [3].

Advantage of high-level language

1. It's fast and efficient, especially if you're working with big data or machine learning algorithms that require many iterations.
2. Fewer lines of code are needed to write applications and software.
3. High-level programming languages can be used to write programs on any computer, including mobile devices, tablets, and personal computers [4].

Disadvantage of high-level language

1. The main disadvantages of high-level languages are that they're less efficient than low-level languages and often don't support all features of modern computer hardware or operating systems.
2. High-Level Languages are not capable of interacting with hardware directly, like low-level languages.
3. The codes written in a high-level language are translated into machine language. Therefore, it takes time to execute [4].

Advantage of low-level language

1. Assembly and Machine Code are Examples of Low-Level Languages.
2. Low-Level Language can directly communicate with computer hardware and other devices.
3. They can manipulate or play with register and storage devices [4].

Disadvantage of low-level language

1. Assembly and Machine Code are difficult to learn.
2. The codes written in Low-Level Language are incredibly tedious to execute.
3. The Low-Level Language is difficult to learn and implement [4].

Application Software

Application software (App) is a kind of software that performs specific functions for the end user by interacting directly with it. The sole purpose of application software is to aid the user in doing specified tasks [3].

Types of application software

Tailored/Custom

custom system software refers to software applications that are specifically designed, developed, and customized to meet the unique requirements and needs of a particular organization or user. This contrasts with off-the-shelf or generic software that is created for a broad user base and may not fully align with the specific processes or workflows of a particular entity [3].

Packaged system software

Packaged system software, also known as off-the-shelf software, refers to pre-built software applications that are commercially available and designed to cater to a wide range of users or organizations without the need for extensive customization. These software packages are typically created by third-party vendors and are ready for use without requiring significant modifications [3].

Software acquisition

Software acquisition refers to the process of obtaining software for use within an organization or by an individual. This process involves selecting, purchasing, or otherwise acquiring software to meet specific needs and requirements. Software acquisition can take various forms, and the methods used depend on factors such as the type of software, licensing models, and the preferences of the acquiring entity. Here are some common methods and considerations in software acquisition [3].

Properties of software acquisition

1. Planning
2. Define product requirements.
3. Determine the acquisition approach.
4. Identify and evaluate potential supplies.
5. Select supplies.
6. Negotiate and contract.

Types of software acquisition

1. Free download
2. Open source
3. Software company
4. Software distributor
5. In house development
6. Outsourced.
7. Packaged

Operating system

An operating system (OS) is a crucial software component that serves as an intermediary between computer hardware and user applications. It provides essential services, manages system resources, facilitates communication between software and hardware, and ensures efficient and secure execution of tasks on a computer or computing device [3].

Objective of Operating system

Two views of OS

OS is an extended machine.

OS basically helps in hiding the complexity and difficulty of the hardware and presents a pleasing and nice-looking interface to the user. And It not only shields the user from the hardware but also hides a lot of unpleasant business concerning interrupts, timers, memory management, and other low-level features. In all the above cases, the notion offered by the operating system is very simple and easy to use than that offered by the hardware. So, here we can say that the function or the work of the operating system is to present the user with the equivalent of an extended machine that is easier to operate or work with as compared to the underlying hardware [5].

OS as a resource manager

Now-a-days all modern computers consist of processors, memory, timers, network interfaces, printers, and so many other devices. The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view. The operating system allows multiple programs to be in memory and run at the same time. Resource management includes multiplexing or sharing resources in two different ways: in time and in space. In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another. In space multiplexing, instead of the customers taking a chance, each one gets part of the resource. For example − Main memory is divided into several running programs, so each one can be resident at the same time [5].

Function of OS

Process management

In a multi-programming environment, the OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called Process Scheduling. An Operating System performs the following activities for Processor Management. An operating system manages the processor's work by allocating various jobs to it and ensuring that each process receives enough time from the processor to function properly. Keeps track of the status of processes. The program which performs this task is known as a traffic controller. Allocates the CPU that is a processor to a process. De-allocates processor when a process is no longer required [5].

Memory management

The operating system manages the Primary Memory or Main Memory. Main memory is made up of a large array of bytes or words where each byte or word is assigned a certain address. Main memory is fast storage, and it can be accessed directly by the CPU. For a program to be executed, it should be first loaded in the main memory. An operating system manages the allocation and deallocation of memory to various processes and ensures that the other process does not consume the memory allocated to one process. An Operating System performs the following activities for Memory Management:

It keeps track of primary memory, i.e., which bytes of memory are used by which user program. The memory addresses that have already been allocated and the memory addresses of the memory that has not yet been used. In multiprogramming, the OS decides the order in which processes are granted memory access, and for how long. It Allocates the memory to a process when the process requests it and deallocates the memory when the process has terminated or is performing an I/O operation [5].

Device management

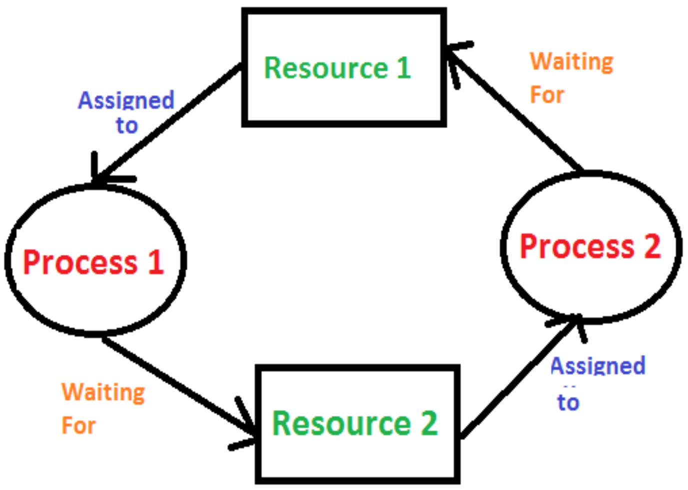
An OS manages device communication via its respective drivers. It performs the following activities for device management. Keeps track of all devices connected to the system. Designates a program responsible for every device known as the Input/Output controller. Decide which process gets access to a certain device and for how long. Allocates devices effectively and efficiently. Deallocates devices when they are no longer required. There are various input and output devices. An OS controls the working of these input-output devices. It receives the requests from these devices, performs a specific task, and communicates back to the requesting process [5].

File management

A file system is organized into directories for efficient or easy navigation and usage. These directories may contain other directories and other files. An Operating System carries out the following file management activities. It keeps track of where information is stored, user access settings, the status of every file, and more. These facilities are collectively known as the file system. An OS keeps track of information regarding the creation, deletion, transfer, copy, and storage of files in an organized way. It also maintains the integrity of the data stored in these files, including the file directory structure, by protecting against unauthorized access [5].

Deadlock management

A deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1 [5].



References

[1] "tech target," [Online]. Available: techtarget.com. [Accessed 21 01 2024].

[2] "Java point," [Online]. Available: https://www.javatpoint.com/. [Accessed 21 01 2024].

[3] "chatgpt," [Online]. Available: chatgpr.com. [Accessed 21 01 2024].

[4] [Online]. Available: https://www.chtips.com/. [Accessed 21 01 2024].

[5] "Greeksfor geeks," [Online]. Available: https://www.geeksforgeeks.org/. [Accessed 21 01 2024].