1.3 - Types of Software

Created By (UMESH KUMAR MEGHWAL)

Our Website Tap this

In general, software can be categorized into two basic types: **System Software** and **Application Software**.

System and Application Software

System Software: System software facilitates easy operation and interaction between the hardware and the user. It acts as an intermediary between the user and the hardware, controlling how the hardware behaves and providing fundamental functions required by the user. It initializes and loads into the system's memory once the computer is powered on, running in the background without direct user interaction.

Examples of System Software:

- **Operating Systems**: Windows, Linux
- **Search Engines**: Chrome

System Software Features:

- Complex design
- Direct connection between hardware and software
- Challenges in manipulation
- Smaller in size
- Difficult to understand
- Typically written in a simple language
- Must be highly efficient

Operating System

An **Operating System (OS)** is the most basic type of system software that controls computer hardware and software, ensuring the effective operation of each computer device. Key OS types include MacOS, Linux, Android, and Microsoft Windows.

Functions of an Operating System:

-
-
- File management
- Main memory management

  CPU management

  Encryption and security
- Process scheduling
- Hardware activation
- Application software management
- Network access
- System use limitations
- Input and output device management
- Software installation and troubleshooting
- System component linking

Programming Language Translators

Programming Language Translators convert high-level languages into machine language, which consists of binary code (0s and 1s). Translators include compilers, interpreters, and assemblers.

Function:

- Converts high-level languages (e.g., Java, Python) into machine code for CPU processing.

Device Drivers

Device Drivers are system software components that manage communication between the operating system and hardware components. Most drivers are pre-installed by the manufacturer.

Function:

- Simplify management and control of hardware.

Firmware Software

Firmware is a type of software installed on computer boards to manage and regulate device operations. It is stored in non-volatile chips like Flash or ROM.

Types:

- **BIOS (Basic Input/Output System)**: Manages booting and hardware checks.
- **UEFI (Unified Extensible Firmware Interface)**: Modern replacement for BIOS.

Utility Software

Utility Software assists in maintaining and optimizing computer systems. It often comes bundled with the operating system and helps manage system performance and security.

Functions:

- Protection from threats
- Disk compression and partitioning
- Data backup and system security
- Disk defragmentation
- Data recovery
- Antivirus protection

Application Software

Application Software refers to programs used by end-users to perform specific tasks, such as internet research, note-taking, graphic design, and more.

Common Features:

- Data manipulation
- Information control
- Calculations
- Visual development
- Resource management
- Report writing
- Spreadsheet creation
- Image editing
- Website development

Examples:

- **Word Processing Software**: Microsoft Word
- **Spreadsheet Software**: Microsoft Excel
- **Presentation Software**: Microsoft PowerPoint
- **Multimedia Software**: VLC Media Player, MX Player
- **Web Browsers**: Chrome, Firefox

-
-
- **Graphics Software**: Adobe Photoshop, PaintShop

Special Categories of Software

Freeware: Free to use, but source code cannot be modified (e.g., Skype).

Shareware: Free to try, but requires payment for continued use (e.g., WinZip).

- **Simulation Software**: Used for monitoring and simulating actions (e.g., MATLAB).
- **Open Source**: Source code is accessible for modification and enhancement (e.g., Linux).
- **Closed Source**: Source code is proprietary and not accessible for modification (e.g., Microsoft Windows).

Business Application Software

Business application software is designed to enhance productivity and efficiency in organizational operations. It includes:

- **Customer Relationship Management (CRM)**: Manages customer interactions and data.
- **Project Management Software**: Aids in scheduling, resource management, and tracking progress.
- **Database Management Systems (DBMS)**: Manages, stores, and retrieves organizational data.

Compilers and Interpreters

Compilers

Definition: A compiler is a specialized computer application that translates the source code of a highlevel programming language (e.g., Java, C++) into machine code, bytecode, or another programming language.

Function:

- Converts the entire source code at once.
- Detects errors after reading the entire code.
- Platform-dependent.
- Slower operating time compared to interpreters due to the need to process the entire code in one go.

Types of Compilers:

1. **Cross-compiler:**

- Operates on one platform (Platform A) and produces executable code for another platform (Platform B).

2. **Source-to-source Compiler:**
    - Translates source code from one programming language to the source code of another language.

C Compiler and Its Versions:

- **Compilation Process:**
    1. **Pre-Processing:** Removes comments, expands macro definitions, and performs file inclusion.
    2. **Compilation:** Converts pre-processed code into assembly-level instructions.
    3. **Assembly:** Translates assembly instructions into machine-level object code.
    4. **Linking:** Connects function calls to their definitions and adds necessary startup and shutdown code.
- **Common C Compilers:**
    - **Turbo C:** Known for its speed and efficiency, released in 1987.
    - **Tiny C Compiler (TCC):** Designed for minimal systems, with a focus on fast compilation.
    - **Portable C Compiler (PCC):** Used widely in the 1970s, known for its validity checks.
    - **GNU Compiler Collection (GCC):** Supports multiple languages, with a strong emphasis on optimization and performance.
    - **Clang:** Part of the LLVM project, supports C, C++, and Objective-C.

Example of C Compilation:

```
#include <stdio.h>

int main(void) {
printf("Hello, World!\n");
return 0;
}
```

Command to compile: gcc main.c Executable output: a.out Run the program: ./a.out **Interpreters**

Definition: An interpreter is a computer program that translates high-level language into machine language, executing it line by line.

Function:

-
-
- Converts and executes statements one at a time.
- Logs errors immediately if they occur.

  No intermediate code generation.

  Typically faster for development and debugging but slower for overall execution compared to compilers.

Types of Interpreters:

1. **Syntax-directed Interpreter:** Uses syntax rules to process input.
2. **Threaded Interpreter:** Executes a sequence of instructions by threading them.
3. **Bytecode Interpreter:** Translates code into bytecode, which is then executed by a virtual machine.

Assembler, Linker, and Loader

Assembler

Definition: An assembler is a tool that translates assembly language code, which uses mnemonics and symbolic names, into machine code that the CPU can execute directly.

Roles and Functions:

- **Representation of Machine Code:** Converts human-readable mnemonics (e.g., ADD, MOV) into binary machine code.
- **Assembly Language Instructions:** Each instruction corresponds directly to a machine code operation, tailored to the computer's architecture.
- **Translation Process:** Converts assembly code into binary machine code through a process called assembling.
- **Registers and Memory:** Provides access to CPU registers and system memory.
- **Direct Mapping to Machine Code:** Each assembly instruction maps one-to-one with machine code instructions.
- **Platform-Specific:** Tailored to specific computer architectures, making code non-portable.
- **Efficiency and Control:** Allows fine-grained control and optimization of hardware, suitable for performance-critical applications.
- **Learning and Teaching:** Serves as a tool for understanding computer architecture and lowlevel programming.

Linker

Definition: A linker is a utility that combines multiple object files generated by the compiler into a single executable program, resolving references between them.

Roles and Functions:

- **Object Files:** Contains machine code and data, produced from source code compilation.
- **Symbol Resolution:** Links function and variable references across object files.
- **Address Binding:** Assigns final memory addresses to variables and functions.
- **Static Linking:** Combines object files and libraries into a standalone executable before runtime.
- **Dynamic Linking:** Links libraries at runtime, allowing the use of shared libraries (DLLs).
- **Library Linking:** Integrates external libraries, resolving references to their functions and variables.
- **Relocation:** Adjusts addresses in object files to match the final memory layout.
- **Symbol Table:** Maintains a table mapping symbol names to memory addresses.
- **Executable Output:** Produces the final executable file or a program ready to use shared libraries.

Loader

Definition: A loader is a system utility that loads an executable file into memory, preparing it for execution by the operating system.

Roles and Functions:

- **Loading Process:** Transfers the executable file from storage (disk) to RAM for execution.
- **Executable File Formats:** Supports various executable formats (e.g., ELF, PE).
- **Address Binding:** Assigns actual memory addresses to program components.
- **Relocation:** Adjusts code and data addresses to reflect their memory location.
- **Dynamic Linking:** Loads shared libraries or DLLs into memory at runtime.
- **Linking Libraries:** Resolves references to external libraries, making their functions available.
- **Initialization:** Sets up the program's stack, initializes global variables, and prepares the execution environment.
- **Program Execution:** Transfers control to the program's entry point to start execution.
- **Loader Types:** Includes absolute loaders, relocatable loaders, and dynamic loaders.
- **Loader Error Handling:** Identifies and manages issues like missing files or address conflicts.