

UNIT 3

Operating System

(4 Hours) — Operating Systems

3.1 Introduction to Operating System

A computer system is broadly made up of four components: the hardware, the operating system, the application programs, and the users. The hardware — the central processing unit (CPU), memory, and input/output devices — provides the basic computing resources. Application programs such as word processors, browsers, and games define the ways in which these resources are used to solve users' computing problems. Sitting between the hardware and the application programs is the Operating System (OS), which controls and coordinates the use of the hardware among the various application programs for the various users.

In simple terms, an Operating System is system software that acts as an intermediary (or interface) between the computer user and the computer hardware. Its central purpose is twofold: to make the computer system convenient to use, and to make the use of the computer hardware efficient.

The Operating System as an Intermediary

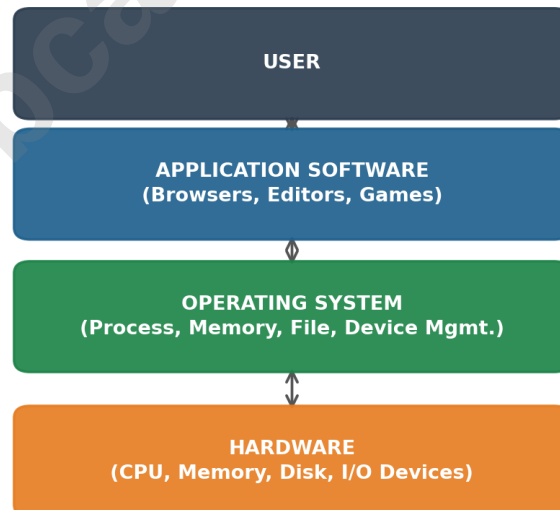


Figure 3.1 — The Operating System acting as an intermediary between hardware and users.

Example 3.1 — Everyday Analogy

Think of the Operating System as the manager of a large office building. The building (hardware) has resources such as rooms, electricity, and lifts. The manager (OS) does not do the actual work of the employees (application programs), but decides who gets which room, when the lift is used, and how electricity is shared — so that everyone can work smoothly without conflict.

3.1.1 Definitions of Operating System

Several equivalent definitions are commonly used to describe an operating system:

- A program that acts as an interface between the user and the computer hardware.
- A resource manager that allocates memory, processor time, and peripheral devices among competing programs.
- A control program that manages the execution of user programs to prevent errors and improper use of the computer.
- An extended machine that hides the messy details of the hardware and presents the user with a clean, easy-to-use set of services.

3.1.2 Why an Operating System is Needed

Without an operating system, every application would need to directly manage the CPU, memory, disks, keyboard, and screen on its own — a task that is both extremely complex and highly error-prone. The OS removes this burden by providing a layer of abstraction: programs simply request services (read a file, allocate memory, print a page) through a well-defined interface, and the OS handles the underlying hardware complexity.

Note: The operating system itself is just a program — but a very special one, since it has direct control over the hardware and grants or denies that control to every other program.

3.2 Function of Operating System

An operating system performs a wide range of functions to keep the computer system running smoothly, efficiently, and securely. These functions can be grouped into the following major categories.

Function	Description
Process Management	Creating, scheduling, suspending, resuming, and terminating processes; allocating CPU time fairly among competing processes.
Memory Management	Allocating and deallocating main memory space to processes; keeping track of which parts of memory are in use and by whom.

Function	Description
File Management	Creating, deleting, reading, writing, and organizing files and directories; controlling access permissions.
Device Management	Managing I/O devices through device drivers; scheduling and controlling access to printers, disks, and other peripherals.
Security & Protection	Preventing unauthorized access to programs and data; authenticating users and controlling privileges.
User Interface	Providing a Command Line Interface (CLI) or Graphical User Interface (GUI) so users can interact with the system.
Error Detection	Monitoring the system for errors in hardware or software and taking corrective action or notifying the user.
Resource Allocation	Fairly and efficiently distributing CPU cycles, memory, and devices among multiple users and jobs.

3.2.1 Process Management

A process is a program in execution. The OS is responsible for creating and deleting processes, suspending and resuming them, and providing mechanisms for process synchronization and communication. It also decides which process gets access to the CPU and for how long, using scheduling algorithms such as First-Come-First-Served (FCFS), Round Robin, or Priority Scheduling.

3.2.2 Memory Management

The OS keeps track of which parts of memory are currently being used and by which process, decides which processes are loaded into memory when space becomes available, and allocates and deallocates memory space as needed. Techniques such as paging and segmentation help the OS use memory efficiently, even allowing programs larger than physical memory to run through virtual memory.

3.2.3 File and Device Management

The file system component organizes data into files and directories, tracks their location on disk, and manages access rights. Device management, meanwhile, uses device drivers to hide hardware-specific details from applications, presenting a uniform interface for very different physical devices such as printers, scanners, and storage drives.

Key Takeaways — Functions of OS

- ✓ The OS manages five core resources: processor, memory, files, devices, and security.
- ✓ Process management handles scheduling and coordination of running programs.
- ✓ Memory management ensures efficient and safe use of RAM, including virtual memory.
- ✓ The OS provides both a user interface (CLI/GUI) and a programming interface (system calls).

bcalink.com

3.3 Types of Operating System

Operating systems have evolved considerably since the earliest computers, adapting to new hardware capabilities and changing user needs. They can be classified into several types based on how they process jobs and manage resources.

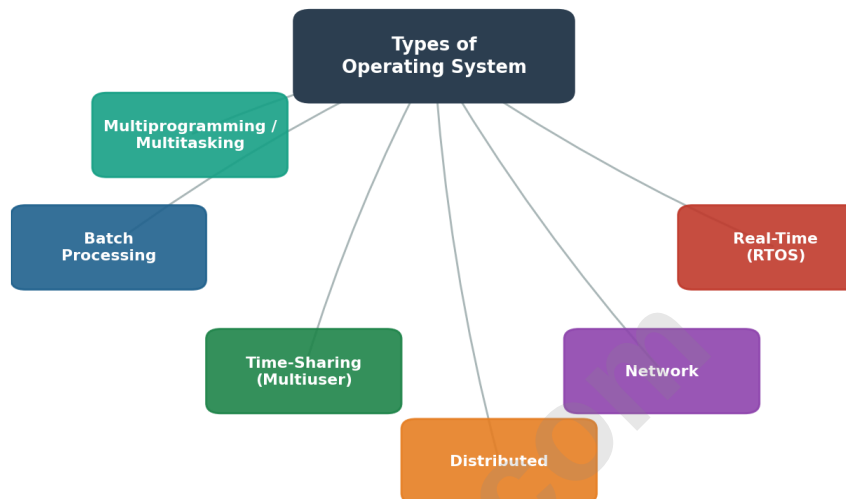


Figure 3.2 — Major categories of operating systems.

Type	Key Characteristic	Example / Use Case
Batch OS	Jobs with similar needs are grouped and executed together with no direct user interaction.	Payroll processing, bank statement generation
Time-Sharing / Multiuser OS	CPU time is divided among multiple users in short time slices, giving the illusion of simultaneous access.	UNIX-based multiuser servers
Distributed OS	Manages a group of independent, networked computers so they appear to users as a single system.	Cluster computing, cloud platforms
Network OS	Runs on a server and provides services such as file sharing and security to computers on a network.	Windows Server, Novell NetWare
Real-Time OS (RTOS)	Guarantees processing of data within a strict, predictable time limit; classified as hard or soft real-time.	Air traffic control, medical devices
Multiprogramming OS	Keeps several jobs in memory simultaneously so the CPU always has a job to execute, improving utilization.	Early mainframe systems

Type	Key Characteristic	Example / Use Case
Multitasking OS	Allows a single user to run several applications at the same time by rapidly switching the CPU between them.	Modern desktop OS (Windows, macOS)

3.3.1 Batch Operating System

In a batch OS, users do not interact directly with the computer. Each user prepares a job (a program along with its data and control information) and submits it to a computer operator. Similar jobs are grouped into batches and executed one after another without user intervention, which improves throughput but offers no interactivity.

3.3.2 Time-Sharing Operating System

Time-sharing (or multitasking, in the multiuser sense) is a logical extension of multiprogramming. The CPU executes multiple jobs by switching among them so quickly that each user gets a very small slice of CPU time in rapid succession, creating the impression that each user has exclusive access to the machine. This provides direct, interactive access to many users simultaneously.

Example 3.2 — Distinguishing OS Types

A university's central server allows 50 students to log in from different labs and run programs at the same time, with each student feeling the machine responds instantly. This is a classic Time-Sharing OS.

A weather satellite's onboard computer must process sensor readings and respond within milliseconds or the data becomes useless. This is a Real-Time OS.

3.3.3 Distributed and Real-Time Operating Systems

A distributed OS manages a collection of physically separate, networked computers so that they appear to the user as a single coherent system, allowing resource sharing and load balancing across machines. A real-time OS, on the other hand, is designed for environments that demand strict timing guarantees. A hard real-time system must meet its deadlines without exception (e.g., an aircraft control system), while a soft real-time system tolerates occasional missed deadlines with reduced quality of service (e.g., video streaming).

Note: Multiprogramming increases CPU utilization by keeping several jobs in memory, while multitasking additionally allows rapid switching so a single user perceives simultaneous execution of applications.

3.4 Open-Source Operating System

An open-source operating system is one whose source code is made publicly available, allowing anyone to view, modify, and redistribute it, usually under a license such as the GNU General Public License (GPL). This is in contrast to proprietary (closed-source) operating systems such as Microsoft Windows, whose source code is kept private by the vendor.

3.4.1 Popular Open-Source Operating Systems

- **Linux** — a family of open-source, UNIX-like kernels first released by Linus Torvalds in 1991; forms the basis of distributions such as Ubuntu, Fedora, and Debian.
- **FreeBSD** — a UNIX-like OS descended from the Berkeley Software Distribution (BSD), widely used in servers and networking appliances.
- **Android (AOSP)** — an open-source, Linux-based OS for mobile devices, developed as the Android Open Source Project.
- **OpenSolaris** — the open-source counterpart of Sun Microsystems' Solaris operating system.

3.4.2 Advantages and Disadvantages

Advantages	Disadvantages
Free to use, modify, and distribute	Steeper learning curve for beginners
Source code can be audited, improving security and transparency	Hardware and software driver support can be inconsistent
Large, active community for support and rapid bug fixes	Commercial technical support may be limited or paid separately
Highly customizable to specific needs	Compatibility issues with some proprietary software

Illustration — Why Servers Prefer Linux

Consider a company hosting a website. Choosing Linux (open-source) over a proprietary OS avoids per-server licensing costs, allows the IT team to inspect the kernel source for security vulnerabilities, and permits deep customization of only the components the server actually needs — reducing both cost and attack surface.

Note: 'Open-source' refers to the availability of source code, not necessarily to the software being free of cost — though most open-source operating systems are also free to use.

Key Takeaways — Unit 3

- ✓ The OS is system software that mediates between hardware and application programs.
- ✓ Its core functions include process, memory, file, and device management, plus security.
- ✓ OS types range from Batch and Time-Sharing to Distributed, Network, and Real-Time systems.
- ✓ Open-source operating systems such as Linux offer transparency, customizability, and no licensing cost, at the expense of a steeper learning curve.

— *End of Unit 3* —

bcalink.com