

OPERATING SYSTEM

Chapter 1 Introduction to Operating System

Study Notes for Engineering Students

Topics Covered

1.1	Operating System, Evolution, Generations - 1st, 2nd, 3rd, 4th
1.2	Mainframe Systems - Batch, Multiprogrammed, Multitasking, Time Sharing, Desktop
1.3	Multiprocessor Systems
1.4	Distributed Systems
1.5	Clustered Systems
1.6	Real Time Systems
1.7	Special-Purpose Systems
1.8	Open-Source Operating System

Practice Section: 15 MCQs with Answer Key and 10 Broad Questions

1.1 Operating System, Evolution and Generations

1.1.1 What is an Operating System?

A computer system is made up of **hardware** (CPU, memory, input/output devices, storage) and **software**. Hardware by itself is only a collection of electronic parts; it cannot do anything useful until it is told what to do. A user, however, does not want to deal with the complicated details of registers, disk controllers and interrupts. The software layer that hides this complexity and makes the machine easy and efficient to use is called the **Operating System (OS)**.

The operating system is the first program loaded into memory when the computer starts (this is known as *booting*) and it keeps running as long as the computer is switched on. All other programs - word processors, compilers, games, browsers - run *under the control* of the operating system and request its services whenever they need to use the hardware.

Definition: An operating system is system software that manages the computer hardware and software resources, acts as an interface between the user and the hardware, and provides an environment in which programs can be executed conveniently and efficiently.

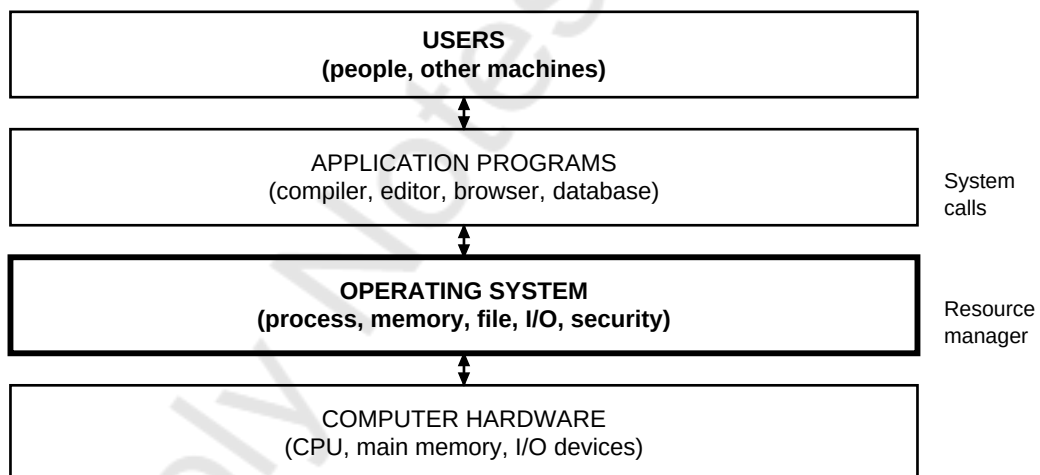


Fig. 1.1: Position of the operating system in a computer system

1.1.2 Two Views of an Operating System

- **OS as a Resource Manager (system view):** The computer has many resources - CPU time, memory space, file storage, I/O devices. Many programs compete for them. The OS decides who gets which resource, when, and for how long, so that the resources are used fairly and efficiently. It acts like a government or a traffic police.
- **OS as an Extended (Virtual) Machine (user view):** The OS presents the user with a simple, clean and easy-to-use abstraction of the hardware. For example, a programmer simply says 'write this data to file X'; the OS handles the disk heads, sectors and controller commands.

1.1.3 Goals of an Operating System

- **Convenience:** make the computer easier to use (especially for desktop and mobile users).
- **Efficiency:** use the computer hardware in an efficient manner (especially for large shared systems).
- **Ability to evolve:** permit the development, testing and introduction of new system functions without interfering with existing services.
- **Protection and fairness:** protect users and programs from each other and share the resources fairly.

1.1.4 Functions of an Operating System

Function	What the OS does
Process management	Creates and deletes processes, schedules the CPU, provides synchronization and communication between processes, handles deadlocks.
Memory management	Keeps track of which parts of memory are in use, allocates and de-allocates memory, swapping, paging and virtual memory.
File management	Creates, deletes, reads, writes files and directories; maps files onto secondary storage; provides backup.
I/O device management	Hides device details through device drivers; provides buffering, caching and spooling.
Secondary-storage management	Free-space management, storage allocation and disk scheduling.
Protection and security	Controls access to resources by users and programs; authentication and permissions.
Networking	Allows communication among computers connected through a network.
Command interpretation	Provides a user interface - a command-line interpreter (shell) or a Graphical User Interface (GUI).
Error detection and accounting	Detects hardware/software errors, keeps logs, and records resource usage by each user.

1.1.5 Kernel and Shell

The **kernel** is the core part of the operating system that always stays in main memory. It directly controls the hardware, handles interrupts, schedules processes and manages memory. The **shell** is the outer layer through which the user talks to the kernel - it can be a text-based command interpreter (such as *bash* or *cmd*) or a graphical desktop. Programs request kernel services through **system calls**.

1.1.6 Evolution of Operating Systems

Operating systems did not appear overnight. They evolved step by step as computer hardware changed and as people demanded better utilization of very expensive machines. The first computers had no operating system at all, and the programmer personally controlled everything. As hardware became faster and cheaper, the idle time of the CPU became a major problem and each new idea - batch processing, multiprogramming, time sharing, personal computing, networking - was introduced to solve a particular shortcoming of the previous approach. This evolution is conveniently studied in terms of computer **generations**.

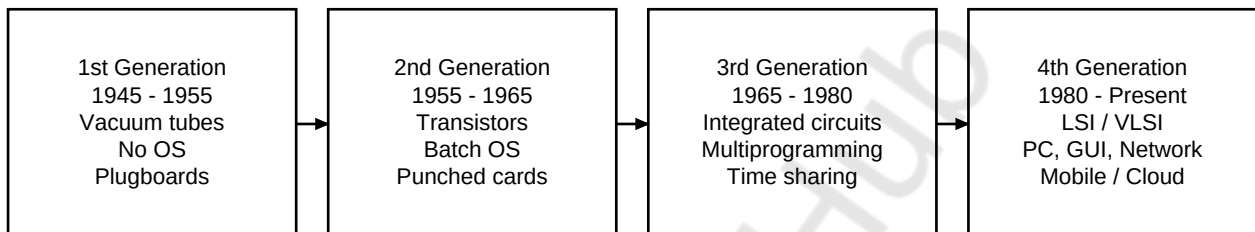


Fig. 1.2: Generations of operating systems

(a) First Generation (1945 - 1955): Vacuum Tubes, No Operating System

The first digital computers (ENIAC, EDVAC, UNIVAC I) were built using **vacuum tubes** and electromechanical relays. They were enormous in size, consumed huge power, produced great heat and failed often. There was **no operating system** and no programming language; programs were written in absolute machine language by wiring **plugboards** or, later, by using punched cards.

- The same group of people designed, built, programmed, operated and maintained the machine.
- A programmer reserved the machine for a fixed time slot using a sign-up sheet and had it entirely to himself (serial access).
- Setup time (loading cards, wiring) was long, and the CPU stayed idle during setup and debugging.
- Used only for simple numerical calculations such as ballistic tables and scientific computations.

(b) Second Generation (1955 - 1965): Transistors and Batch Systems

The invention of the **transistor** made computers smaller, faster and more reliable. These machines (IBM 1401, IBM 7090/7094) were called **mainframes** and were kept in air-conditioned rooms. Programs were now written in assembly language and in high-level languages such as **FORTRAN** and COBOL, punched on cards and submitted to an operator. To reduce wasted setup time, jobs with similar needs were collected into **batches**, and a small program called the **resident monitor** automatically transferred control from one job to the next.

- First true operating systems appeared: FMS (Fortran Monitor System) and IBSYS for the IBM 7094.
- Roles were separated - designers, programmers and professional operators.

- Input/output was done off-line using magnetic tapes (cheap IBM 1401 for card-to-tape and tape-to-printer work, expensive 7094 for computation).
- No interaction between the user and the job; the user got the result hours or days later (long turnaround time).

(c) Third Generation (1965 - 1980): Integrated Circuits and Multiprogramming

Integrated circuits (ICs) replaced individual transistors, giving much higher speed and lower cost. IBM introduced the **System/360** family of compatible computers with a single large operating system, **OS/360**. The main new ideas of this generation were **multiprogramming** (keeping several jobs in memory to keep the CPU busy), **spooling** (reading jobs from cards to disk while another job runs) and **time sharing** (interactive use by many users through terminals).

- Important systems: OS/360, CTSS, MULTICS (MIT, Bell Labs, GE) and the first version of **UNIX** (Ken Thompson and Dennis Ritchie, Bell Labs, 1969).
- Minicomputers such as the DEC PDP-11 appeared; disks replaced tapes as the main storage.
- Introduction of virtual memory, file systems, device independence and the concept of a process.
- Much better CPU utilization and throughput; users could work on-line through terminals.

(d) Fourth Generation (1980 - Present): Personal Computers, GUI and Networking

Large Scale Integration (LSI) and **Very Large Scale Integration (VLSI)** placed thousands to millions of transistors on a single chip, leading to the microprocessor and the **personal computer (PC)**. The computer was no longer a shared resource; it became a personal tool, so operating systems had to be *user friendly*.

- Early PC systems: CP/M and **MS-DOS** (1981, command-line, single user, single task).
- **Graphical User Interface (GUI)**: Apple Macintosh (1984), Microsoft Windows (1985 onwards, Windows NT 1993).
- UNIX workstations, and the open-source **Linux** kernel (Linus Torvalds, 1991).
- **Network operating systems** and **distributed operating systems** for connected computers; client-server computing.
- Multiprocessor and multi-core support, real-time and embedded operating systems.
- Modern mobile operating systems (iOS, Android) and cloud/virtualization-based systems.

Summary of Generations

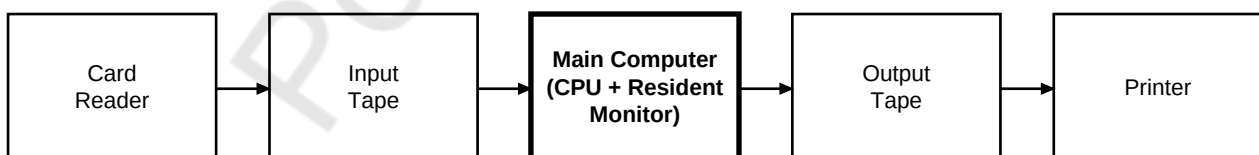
Generation	Period	Hardware	OS / Software	Examples and Features
First	1945 - 55	Vacuum tubes, relays	No OS; machine language	ENIAC, UNIVAC I; plugboards; serial access; one user at a time
Second	1955 - 65	Transistors	Batch operating systems; FORTRAN, COBOL	IBM 1401, 7094; FMS, IBSYS; punched cards; resident monitor
Third	1965 - 80	Integrated circuits (SSI/MSI)	Multiprogramming, spooling, time sharing	IBM System/360 (OS/360), CTSS, MULTICS, UNIX; disks; virtual memory
Fourth	1980 - now	LSI / VLSI microprocessors	PC, GUI, network and distributed OS, mobile OS	MS-DOS, Windows, macOS, Linux, Android, iOS; networking; multitasking

1.2 Mainframe Systems

Mainframe computers were the first computers used to tackle many commercial and scientific applications. Mainframe operating systems developed in stages - from the simple **batch system**, to the **multiprogrammed system**, and finally to the **time-sharing system**. Ideas born on the mainframe (multitasking, protection, virtual memory) later moved to desktop operating systems. This section studies each stage in turn.

1.2.1 Batch Systems

In early computers, a programmer prepared a program and data on **punched cards** and handed it to a computer operator. The operator then loaded the cards, ran the job and returned the printed output. Because the setup for each job took a long time, the operator **grouped jobs having similar requirements into a batch** (for example, all FORTRAN jobs) and ran the whole batch together. The OS simply executes the jobs in the batch **one after another**, automatically, without any intervention from the user. This is the earliest form of operating system.



Jobs of a similar type are batched, loaded on tape, and run one after another without any interaction between the user and the running job.

Fig. 1.3: Working of a batch processing system

Working of a Batch System

1. Users prepare their jobs (program + data + control cards written in **Job Control Language, JCL**) and submit them to the operator.

2. The operator sorts the jobs into batches of similar type and loads them on an input device (card reader or tape).
3. The **resident monitor** - a small program that always stays in a fixed area of memory - loads one job at a time into the user area.
4. When the job finishes (or fails), control returns to the monitor, which automatically loads the next job.
5. Output of every job is collected on an output device and returned to the respective user.

Spooling (*Simultaneous Peripheral Operations On-Line*) later improved batch systems: instead of working directly with slow card readers and printers, jobs are first read onto a disk, executed from the disk, and the output is stored on the disk until the printer is ready. This overlaps the I/O of one job with the computation of another.

Features and Characteristics	Advantages	Disadvantages
No user interaction during execution. Jobs with similar needs are batched. Sequential execution (FCFS). Resident monitor, JCL. Memory holds the monitor and one job at a time.	Reduces setup time per job. Easy to manage a large number of similar jobs. Repetitive jobs (payroll, billing, bank statements) run without human attention. Can be run at off-peak hours.	Long turnaround time. No interaction - debugging is difficult. CPU idle when the running job waits for I/O (CPU is much faster than I/O devices). One job error may delay all other jobs. No priority handling.

1.2.2 Multiprogrammed Systems

A single user cannot keep the CPU and the I/O devices busy all the time. In a batch system, whenever the running job asks for an I/O operation, the CPU sits idle until the I/O is completed. **Multiprogramming** removes this waste. Several jobs are kept in **main memory at the same time**. The OS picks one job and begins executing it. When this job has to wait (for a tape to be mounted, an I/O operation to finish and so on), the OS simply **switches the CPU to another job**. When that job waits, the CPU switches to a third, and so on. As long as at least one job is ready to execute, the CPU is never idle.

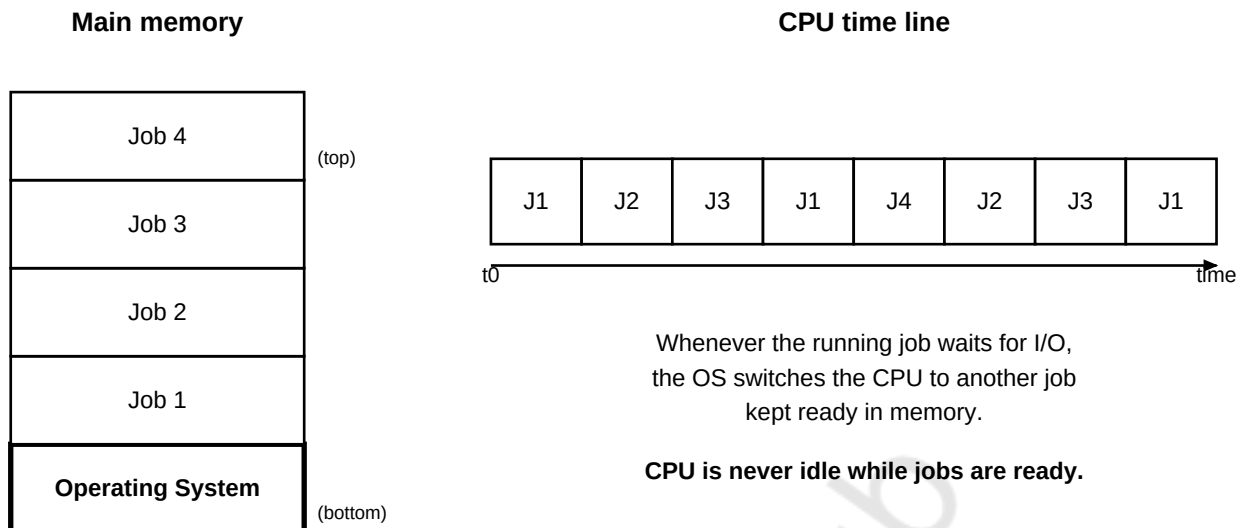


Fig. 1.4: Memory layout and CPU switching in a multiprogrammed system

Functions Required for Multiprogramming

- **Job scheduling (long-term scheduling):** selecting which jobs from the job pool on disk are to be brought into memory.
- **CPU scheduling (short-term scheduling):** deciding which ready job in memory gets the CPU next.
- **Memory management:** allocating memory to several jobs and keeping them separate.
- **I/O management and protection:** one job must not damage the OS or another job.
- **Deadlock handling and synchronization** among competing jobs.

CPU Utilization

If a process spends a fraction p of its time waiting for I/O and n processes are in memory, the probability that all n processes wait simultaneously is p^n . Hence, **CPU utilization** = $1 - p^n$. The number n is called the **degree of multiprogramming**. Example for $p = 0.8$ (80% I/O wait):

Degree of multiprogramming (n)	1	2	4	6	10
CPU utilization	20%	36%	59%	74%	89%

Advantages	Disadvantages
High CPU utilization and throughput. Reduces CPU idle time. Better use of memory and I/O devices. More jobs completed in less time.	Complex scheduling and memory management. Jobs are still non-interactive. Protection between jobs is required. Long jobs may delay short jobs; no guarantee on response time.

1.2.3 Multitasking Systems

Multitasking is the logical extension of multiprogramming. The CPU executes **multiple tasks (processes)** by **switching among them so rapidly** that the user feels that all the tasks are running at the same time, and the user can interact with each program while it is running. A single-CPU computer actually executes only one task at any instant; the illusion of parallelism is created by fast switching, called **context switching**. Modern desktop operating systems such as Windows, Linux, macOS and Android are multitasking systems; the user can, for example, listen to music, download a file and type a document simultaneously.

Basis	Cooperative Multitasking	Preemptive Multitasking
Control of CPU	Running task gives up the CPU voluntarily	OS forcibly takes the CPU back using a timer interrupt
Risk	A faulty or greedy task can hang the whole system	A faulty task cannot monopolise the CPU
Responsiveness	Poor	Good
Examples	Windows 3.x, classic Mac OS	Windows NT onwards, UNIX/Linux, macOS, Android

- Multitasking is mostly used in interactive systems; it needs a good CPU scheduler and memory protection.
- **Multithreading** is a finer form of multitasking in which several threads of the same program run concurrently.
- **Difference from multiprogramming:** multiprogramming switches mainly when a job waits for I/O (to improve CPU utilization), while multitasking switches frequently (also on time expiry) to improve user responsiveness.

1.2.4 Time-Sharing Systems

A **time-sharing (multi-user) system** allows many users to share the computer at the same time, each working through his or her own terminal. The CPU is allocated to each user in turn for a very short period called a **time slice or time quantum** (typically 10 to 100 milliseconds). When the quantum expires the OS preempts the program and gives the CPU to the next user, usually in a **round-robin** manner. Because the switching is so quick, each user gets the impression of having the whole machine. The objective is to **minimize response time** (usually less than one second) rather than to maximize CPU utilization.

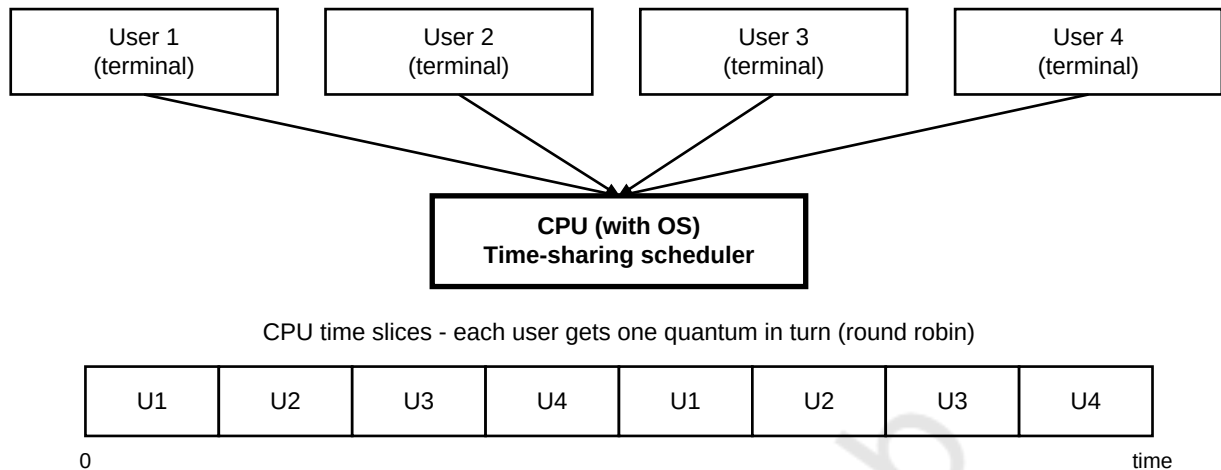


Fig. 1.5: Time-sharing system with round-robin CPU allocation

Requirements of a Time-Sharing System

- Interactive file system on-line and an interactive command interpreter.
- CPU scheduling with a time quantum (round robin).
- Memory management and **virtual memory** and **swapping** to move processes between memory and disk.
- Process synchronization and communication; deadlock avoidance.
- Protection and a login/authentication mechanism for every user.

Examples: CTSS, MULTICS, UNIX, IBM TSS/360 and VAX/VMS. UNIX and Linux servers with multiple login sessions are time-sharing systems.

Advantages	Disadvantages
Quick response; users interact directly with the system. Reduces CPU idle time. Many users share costly hardware (cost effective). Avoids duplication of software; sharing of data and programs.	Needs more complex OS (scheduling, memory, protection). Overhead of frequent context switching. Security and privacy of users' data must be ensured. Reliability issues - if the main computer fails all users are affected.

1.2.5 Desktop Systems

Personal computers (PCs) appeared in the 1970s and became **desktop systems**. A desktop system is a computer meant for use by **one person at a time**, with I/O devices such as keyboard, mouse, monitor and printer. In the beginning, their operating systems (such as MS-DOS) had no file or memory protection, because only one user used the computer and CPU utilization was not important. The goals of a desktop OS are therefore **user convenience and responsiveness** rather than maximum CPU utilization.

As hardware became powerful, desktop operating systems adopted the techniques developed for mainframes - multitasking, virtual memory, file protection, networking and multi-user accounts.

Today they are generally connected to the Internet and local networks and are exposed to threats such as viruses and malware, so security is an important feature.

- **Examples:** MS-DOS, Microsoft Windows family, Apple macOS, Linux distributions (Ubuntu, Fedora).
- **Features:** GUI with windows, icons and menus; plug-and-play devices; multitasking; networking; multimedia support; user accounts and file permissions.
- **Types of desktop machines:** desktop PC, laptop/notebook, workstation.

Comparison of Mainframe Operating System Types

Basis	Batch	Multiprogrammed	Multitasking / Time Sharing	Desktop
Main goal	Reduce setup time, automate job sequencing	Maximize CPU utilization	Minimize response time	User convenience
User interaction	None	Very little	Interactive	Interactive (GUI)
Jobs in memory	One	Several	Several (with time slices)	Several
CPU switching	Only at job end	When job waits for I/O	At I/O wait and on time-quantum expiry	As in multitasking
Turnaround / Response	Very long	Long	Short	Short
Examples	FMS, IBSYS	OS/360 (MFT/MVT)	UNIX, MULTICS, CTSS	MS-DOS, Windows, macOS

1.3 Multiprocessor Systems

Most systems studied so far use a single general-purpose processor. A **multiprocessor system** (also called a **parallel system** or **tightly coupled system**) has **two or more processors in close communication**, sharing the computer bus, the system clock, and sometimes memory and peripheral devices. A single operating system controls all the processors, and communication between them takes place through shared memory.

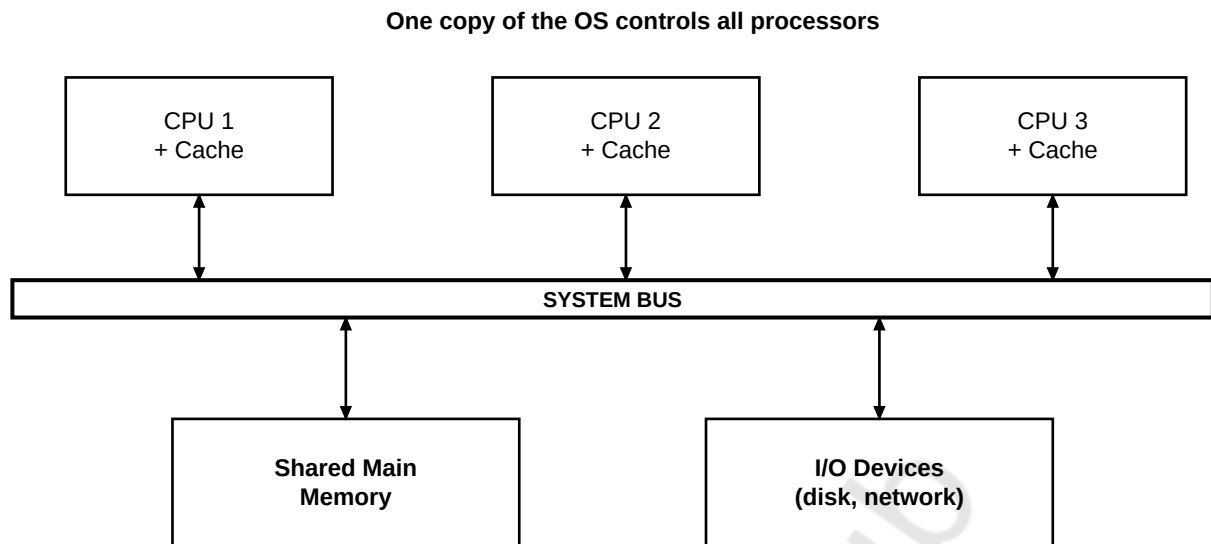


Fig. 1.6: Architecture of a symmetric multiprocessor (SMP) system

1.3.1 Advantages of Multiprocessor Systems

Advantage	Explanation
Increased throughput	More processors give more work done in less time. The speed-up with N processors is, however, less than N , because of the overhead of keeping all processors working correctly and contention for shared resources.
Economy of scale	Cheaper than several single-processor systems, since peripherals, storage and power supplies are shared. Data is shared and stored on one disk.
Increased reliability	If one processor fails, the system does not halt; the remaining processors share its work and the system slows down. This is called graceful degradation . Systems that continue even after failure are fault tolerant (e.g., Tandem NonStop).

Amdahl's law: if a fraction **f** of a program can be parallelised and N processors are used, the speed-up is $1 / ((1 - f) + f/N)$. For $f = 0.9$ and $N = 4$, speed-up = $1 / (0.1 + 0.225) = 3.08$, not 4.

1.3.2 Types of Multiprocessing

Asymmetric Multiprocessing (AMP) - each processor is assigned a specific task. One **master processor** controls the system and allocates work to the **slave processors**, which wait for instructions or have predefined tasks (master-slave relationship). **Symmetric Multiprocessing (SMP)** - each processor runs an identical copy of the operating system and all processors are **peers**; no master-slave relationship exists. Windows, Linux, macOS and modern UNIX versions support SMP.

Basis	Asymmetric Multiprocessing	Symmetric Multiprocessing
Relationship	Master - slave	Peer to peer; all processors are equal
OS execution	Only the master runs the OS and does scheduling	Every processor can run the OS code
I/O handling	Usually by the master only	Any processor can perform I/O

Basis	Asymmetric Multiprocessing	Symmetric Multiprocessing
Load balancing	Poor; master may become a bottleneck	Good; work is shared evenly
Reliability	Failure of the master stops the system	Failure of one processor does not stop the system
Complexity	Simple	More complex (synchronization of kernel data structures)
Examples	Early SunOS 4, some embedded systems	Windows, Linux, Solaris, macOS

1.3.3 Other Related Concepts

- **Multi-core processors:** several processing cores on the same chip; communication on-chip is faster and uses less power than between chips. Used in all modern PCs and smartphones.
- **UMA (Uniform Memory Access):** all processors take the same time to access any memory location (typical SMP).
- **NUMA (Non-Uniform Memory Access):** each processor has local memory that is faster to access than the memory of another processor; used for larger systems.
- **Blade servers:** multiple processor boards, I/O boards and networking boards placed in the same chassis, each board running its own OS.

Advantages	Disadvantages
High performance and throughput. Graceful degradation and reliability. Economy of scale. Parallel processing of large problems.	Complicated OS design (scheduling, synchronization). Needs special hardware and larger main memory. Cost is higher than uniprocessor systems. Bus contention limits the number of processors.

1.4 Distributed Systems

A **distributed system** is a collection of **independent computers (sites or nodes)** that are connected through a communication network and appear to the user as a **single coherent system**. Unlike a multiprocessor system, the processors **do not share memory or a clock**; each processor has its own local memory and runs its own operating system. They communicate with each other by **message passing** over lines such as high-speed buses, telephone lines, fibre, wireless links, LAN or WAN. For this reason they are called **loosely coupled systems**.

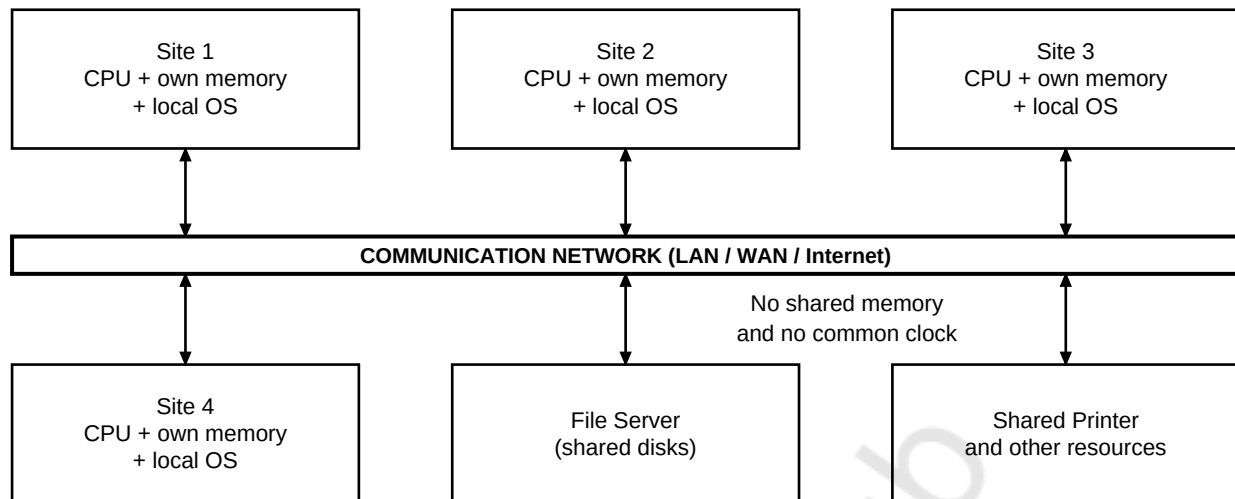


Fig. 1.7: A distributed system

1.4.1 Reasons for Building Distributed Systems

- **Resource sharing:** a user at one site can use the resources (files, printers, databases) of another site.
- **Computation speed-up:** a big computation can be divided into sub-computations that run concurrently on different sites; load sharing moves jobs to lightly loaded sites.
- **Reliability:** if one site fails the remaining sites can continue to work; failure detection and recovery are possible.
- **Communication:** users at different sites can exchange messages, e-mail, files and video.
- **Scalability and economy:** a group of small cheap computers can give more performance per cost than one big mainframe.

1.4.2 Types of Distributed Systems / Models

Model	Description	Example
Client - server	Server systems satisfy requests generated by client systems. <i>Compute-server</i> executes actions requested by clients; <i>file-server</i> provides an interface to store and retrieve files.	Web server and browser, database server, file server
Peer-to-peer (P2P)	All nodes are equal; each node can act as client, server or both. No central server. Joins the network by registering with a lookup service or by broadcasting requests.	BitTorrent, blockchain networks
Network operating system	Users are aware of multiple machines; remote login and file transfer are explicit.	Windows Server, UNIX with NFS
Distributed operating system	Users are not aware of the number or location of machines; the OS manages everything transparently.	Amoeba, Mach, Sprite, LOCUS

1.4.3 Network OS versus Distributed OS

Basis	Network Operating System	Distributed Operating System
Awareness	Users know that many computers are connected	Users see one single system
Transparency	Low	High (location, migration, replication, concurrency, failure)
Autonomy	Each machine is independent and has its own OS	Operating system is common for all, giving a single-system image
Communication	File sharing, remote login	Message passing, remote procedure call, shared objects
Fault tolerance	Less	More
Examples	Novell NetWare, Windows Server, UNIX	Amoeba, Mach, Plan 9

1.4.4 Transparency in Distributed Systems

- **Access transparency:** local and remote resources are accessed in the same way.
- **Location transparency:** users need not know where a resource is physically located.
- **Migration transparency:** resources can move without changing their names.
- **Replication transparency:** users are unaware that several copies of a resource exist.
- **Concurrency transparency:** several users can use shared resources without interfering.
- **Failure transparency:** the system hides the failure and recovery of components.

Advantages	Disadvantages
Resource sharing and load balancing. High reliability and availability. Better performance/cost ratio. Easy to add new sites (scalable). Fast communication among users.	Complex software; difficult design and debugging. Network failure or delay affects the system. Security problems - data travels over the network. Higher set-up and maintenance cost. No global clock; synchronization is difficult.

1.5 Clustered Systems

A **clustered system** gathers together **multiple CPUs (computers) to accomplish computational work**, like a parallel system. It differs from a multiprocessor system in that it is composed of **two or more individual systems - called nodes - coupled together**, each node being a complete computer (single-processor or multiprocessor, with its own OS and memory). The generally accepted definition is that clustered computers **share storage** and are **closely linked via a LAN** or a faster interconnect such as InfiniBand. Clustering is usually used to provide **high availability** service, i.e. service continues even if one or more systems in the cluster fail.

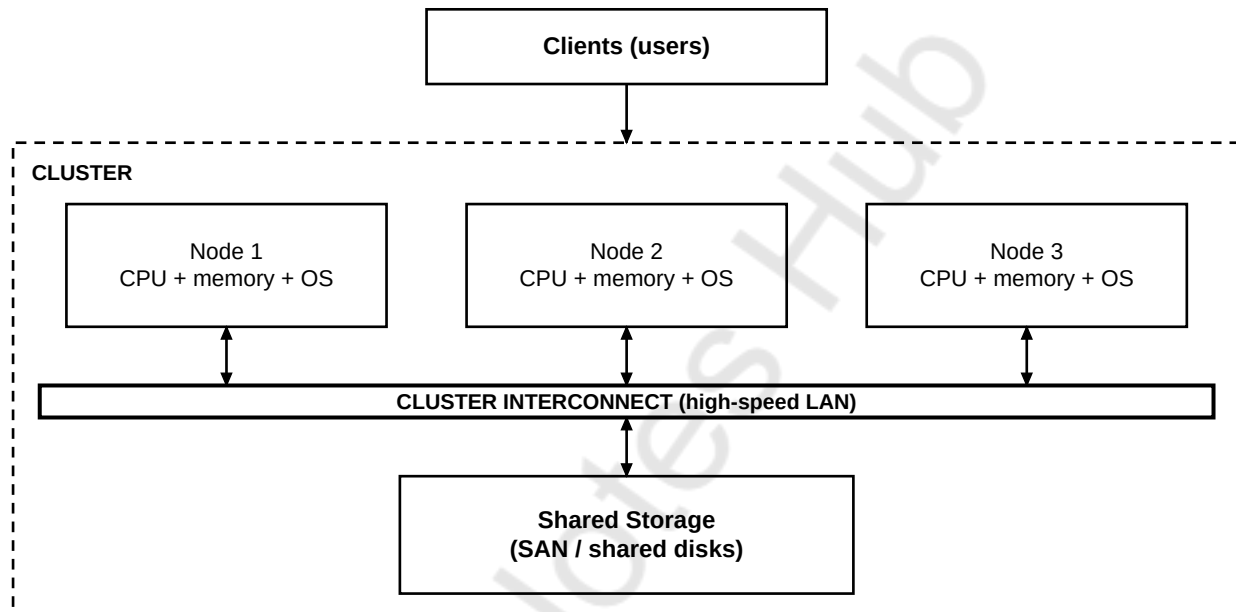


Fig. 1.8: Structure of a clustered system

1.5.1 Types of Clustering

Type	Working
Asymmetric clustering	One machine is in hot-standby mode while the other runs the applications. The hot-standby host only monitors the active server; if that server fails, the hot-standby host becomes the active server.
Symmetric clustering	Two or more hosts run applications and monitor each other . This is more efficient since it uses all available hardware, but requires that more than one application be available to run.
Parallel clustering	Allows multiple hosts to access the same data on the shared storage . A special software such as the Distributed Lock Manager (DLM) is used to avoid conflicting operations. Used for databases (e.g., Oracle Real Application Cluster).
Clustering over a WAN	Clusters at geographically separate locations for disaster recovery.

1.5.2 Classification by Purpose

- **High-availability (HA) clusters:** redundant nodes remove the single point of failure (web servers, banks, databases).

- **Load-balancing clusters:** incoming requests are distributed among the nodes (large websites).
- **High-performance computing (HPC) clusters:** an application is divided into tasks that run on all nodes at the same time - e.g. weather forecasting, simulations. A well-known example is the **Beowulf cluster**, which uses ordinary PCs with Linux.
- **Storage Area Network (SAN):** allows many systems to attach to a pool of storage, so that a failed application can be restarted on another host.

1.5.3 Comparison: Multiprocessor, Distributed and Clustered Systems

Basis	Multiprocessor	Distributed	Clustered
Coupling	Tightly coupled	Loosely coupled	Moderately / closely linked nodes
Memory	Shared memory	No shared memory (each site own memory)	Own memory; shared storage (disks)
OS	One OS controls all CPUs	Each site has its own OS (common OS in DOS)	Each node has its own OS plus cluster software
Connection	System bus	LAN / WAN	High-speed LAN / interconnect
Main purpose	High throughput, reliability	Resource sharing, communication	High availability, performance
Geographical spread	Single machine	Wide (anywhere)	Usually same room / building

Advantages	Disadvantages
High availability; fault tolerance. Scalability - nodes can be added. Load balancing and high performance at low cost.	Needs cluster management software; complex to configure. Cost of interconnects and shared storage. Shared storage may become a bottleneck / single point of failure.

1.6 Real-Time Systems

A **Real-Time Operating System (RTOS)** is used when there are **rigid time requirements** on the operation of a processor or the flow of data. In such a system the **correctness of the result depends not only on its logical correctness but also on the time at which it is produced**. A real-time system is often used as a control device in a dedicated application - for example controlling scientific experiments, medical imaging systems, industrial control systems, automobile engine fuel-injection systems, home appliances, weapon systems and flight control. A real-time system must respond to external events within a **fixed deadline**.

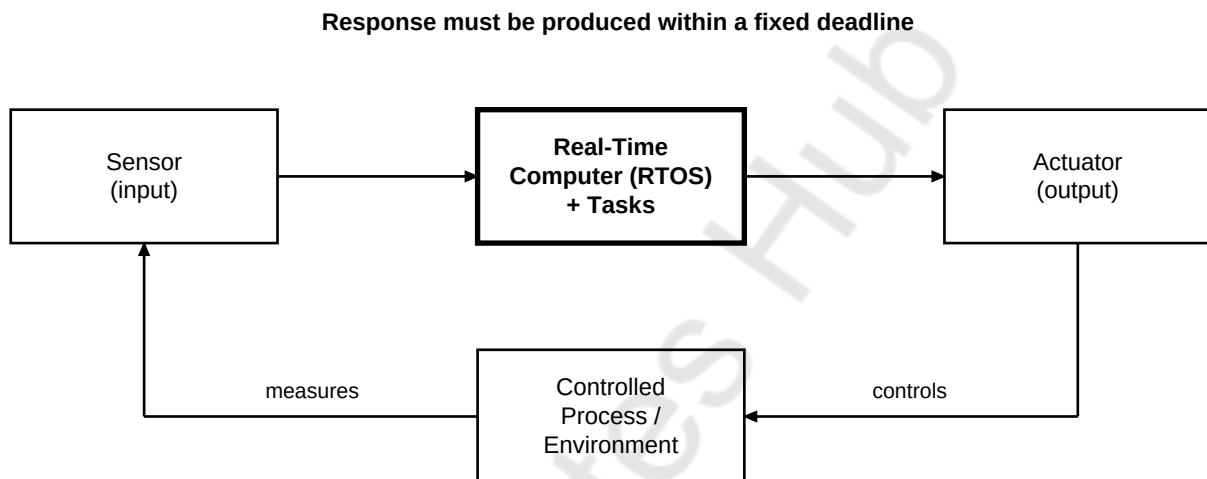


Fig. 1.9: Block diagram of a real-time control system

1.6.1 Types of Real-Time Systems

Basis	Hard Real-Time System	Soft Real-Time System	Firm Real-Time System
Deadline	Must be met absolutely, always guaranteed	Deadline is important but may occasionally be missed	Missing is tolerated rarely; late result is useless
Effect of missing the deadline	Catastrophic failure - loss of life or property	Reduced quality or performance only	Result is discarded; degradation in quality
Secondary storage / virtual memory	Limited or absent (causes unpredictable delays)	Supported	Limited
Response time	Very short and guaranteed	Short, average-based	Short
Examples	Air-traffic control, pacemaker, airbag, missile guidance, nuclear plant control	Multimedia streaming, online gaming, video conferencing, virtual reality	Assembly-line robots, video frame decoders

1.6.2 Characteristics and Features of an RTOS

- **Determinism:** operations are performed at fixed, predetermined times or within predetermined intervals.

- **Responsiveness:** very small interrupt latency and fast context switch.
- **Priority-based preemptive scheduling:** a high-priority task always preempts a low-priority task. Common algorithms are **Rate Monotonic Scheduling (RMS)** and **Earliest Deadline First (EDF)**.
- **Small kernel (micro-kernel) and small memory footprint;** usually no complex user interface.
- **Reliability and fail-safe operation:** the system must continue working even in the case of partial failure.
- Support for timers, clocks and inter-task communication (semaphores, message queues, mailboxes).
- Low jitter - the variation in the time of execution of the same task is small.

1.6.3 RTOS versus General-Purpose OS

Basis	Real-Time OS	General-Purpose OS (Windows / Linux desktop)
Design goal	Meet deadlines (predictability)	Maximum throughput and user convenience
Scheduling	Priority-based, deadline-driven	Fairness-oriented time sharing
Latency	Low and bounded	Variable, unbounded
Size	Small kernel	Large
Examples	VxWorks, QNX, RTLinux, FreeRTOS, eCos, LynxOS	Windows, Ubuntu, macOS

Advantages	Disadvantages
Accurate and timely response. Efficient use of system resources; very reliable. Suitable for embedded and mission-critical applications.	Limited multitasking; few tasks can run at a time. Complex and costly to design; difficult to test timing. Heavy use of system resources by the scheduler. Thread priority must be carefully set; low-priority tasks may starve.

1.7 Special-Purpose Systems

The systems discussed so far are general-purpose computer systems - desktops, servers and mainframes. There are also **special-purpose systems** designed to perform a **specific, limited set of functions**. Their operating systems are generally small, with limited features, and are often **customised for the application**. Three classes are important: real-time embedded systems, multimedia systems and handheld systems. (Real-time systems have been discussed in 1.6.)

1.7.1 Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence. They are found in devices such as car engines, manufacturing robots, DVD players, washing machines, microwave ovens, digital cameras, ATMs, smart TVs, medical devices and the Internet of Things (IoT). They have very specific tasks; the underlying systems are primitive, and the operating system provides limited features. Usually they have little or no user interface, preferring to spend their time monitoring and managing hardware devices. Embedded systems almost always run **real-time operating systems**, which may be a general-purpose OS (e.g. embedded Linux), a specialised RTOS (e.g. VxWorks, FreeRTOS) or application-specific code with no OS.

1.7.2 Multimedia Systems

Most operating systems are designed for conventional data such as text files, programs and spreadsheets. A growing area is **multimedia data** - audio, video and animation - with applications such as video on demand, MP3 players, video conferencing and live webcasts. Multimedia files require **continuous-media delivery at a fixed rate** (e.g. 24-30 frames per second for video). The OS must therefore provide **quality-of-service (QoS)** guarantees: fast disk scheduling, timely CPU scheduling, large buffers, data compression and decompression, and high-bandwidth networking. Multimedia systems are usually classified as soft real-time systems.

1.7.3 Handheld Systems

Handheld systems include Personal Digital Assistants (PDAs), smartphones, tablets, e-book readers and smart watches. Developers face challenges because of the **limited size** of such devices:

- **Small memory:** from some hundred kilobytes in early devices to a few gigabytes now - the OS must use memory very efficiently, returning unused memory to the system.
- **Slow processors** (compared to PCs) to save power, since a faster processor needs a bigger battery and generates heat.
- **Small display screens** - web pages must be reformatted and scaled; touch input and voice input are used.
- **Battery life** - the OS must follow strict power-management policies.
- **Wireless connectivity** - Wi-Fi, Bluetooth, mobile data; GPS, camera, sensors.

Examples of mobile operating systems: Android (Google), iOS (Apple), HarmonyOS, KaiOS; older ones - Symbian, Windows Phone, Palm OS.

1.7.4 Other Special-Purpose Systems

System	Purpose and OS features	Examples
Embedded system	Dedicated function, real-time, small footprint, no or minimal UI	VxWorks, FreeRTOS, embedded Linux
Handheld / mobile	Touch UI, power management, wireless, apps	Android, iOS
Multimedia	Continuous media, QoS, compression	Media servers, set-top box OS
Wireless sensor nodes	Tiny memory, event-driven, low power	TinyOS, Contiki
Smart-card system	Very small OS, security, cryptography	Java Card, MULTOS
Automotive systems	Safety-critical, real-time	QNX, AUTOSAR OS

Features of Special-Purpose Operating Systems

Limited resources (memory, CPU, power); small kernel size.
Dedicated to a specific task and customised to the application.
Usually real-time and highly reliable.
Often stored in ROM / flash memory; booting is very fast.
Little or no user interface; remote or minimal control.

1.8 Open-Source Operating System

Open-source software is software whose **source code is made available to the public** under a licence that allows anyone to **view, study, modify and redistribute** it. An **open-source operating system** is an operating system whose source code is freely available. In contrast, closed-source (proprietary) operating systems such as Microsoft Windows and Apple macOS are released only in binary form, and the source code is kept secret and protected by copyright.

1.8.1 Free Software and Open Source - Background

At the beginning of the computing era, software was shared freely. In the 1980s companies started selling software in binary form only. In 1983 **Richard Stallman** started the **GNU Project** to create a free, UNIX-compatible operating system, and in 1985 founded the **Free Software Foundation (FSF)**. He defined four freedoms: to *run* the program, to *study and change* it, to *redistribute copies*, and to *distribute modified versions*. The GNU General Public License (GPL) was created to protect these freedoms ('copyleft'). In 1991 the Finnish student **Linus Torvalds** wrote the **Linux kernel**; combined with the GNU tools it gave a complete free operating system. The term 'open source' was introduced in 1998 and the **Open Source Initiative (OSI)** was founded.

1.8.2 Popular Open-Source Operating Systems

Operating System	Description
Linux (many distributions)	Kernel by Linus Torvalds; distributions include Ubuntu, Debian, Fedora, Red Hat Enterprise Linux, CentOS/Rocky, SUSE, Linux Mint, Arch. Powers servers, supercomputers, embedded devices.
BSD family	FreeBSD, NetBSD, OpenBSD - descended from Berkeley UNIX (BSD); permissive BSD licence.
Android (AOSP)	Mobile OS based on a modified Linux kernel; the Android Open Source Project is open source.
Solaris derivatives	Sun Microsystems released OpenSolaris; continued as illumos / OpenIndiana.
Others	Chromium OS, ReactOS, Haiku, FreeRTOS, Zephyr RTOS, Minix

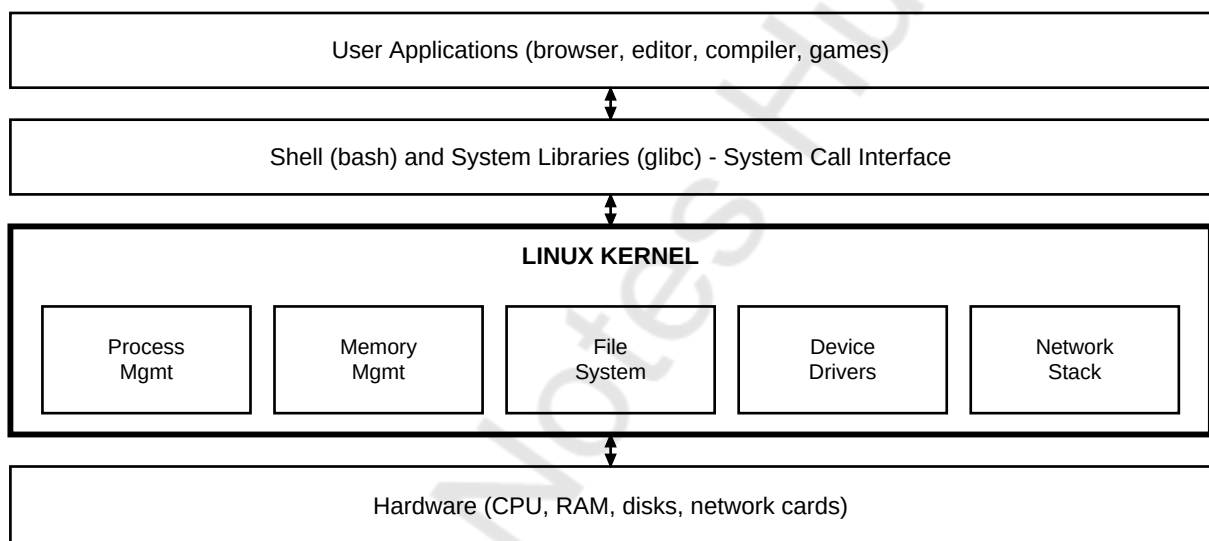


Fig. 1.10: Layered architecture of the Linux operating system

1.8.3 Common Open-Source Licences

Licence	Type	Main condition
GNU GPL	Strong copyleft	Derived works must also be released under the GPL with source code (Linux kernel uses GPL v2)
GNU LGPL	Weak copyleft	Library can be linked with proprietary programs; changes to the library must be shared
BSD (2/3-clause)	Permissive	Can be used even in closed-source products; copyright notice must be kept
MIT	Permissive	Very few restrictions; keep the copyright and licence notice
Apache 2.0	Permissive	Like MIT, plus an explicit patent grant; used in Android userland

1.8.4 Open Source versus Closed Source

Basis	Open-Source OS	Closed-Source (Proprietary) OS
Source code	Available to everyone	Kept secret by the vendor
Cost	Usually free (support may be paid)	Licence fee is required
Modification	Anyone can modify and redistribute	Only the owner can modify
Support	Community, forums and commercial vendors	Official vendor support
Security	Many reviewers can find and fix bugs quickly (but attackers can also read the code)	Security by obscurity; fixes only by the vendor
Ease of use / hardware support	Depends on distribution; improving constantly	Usually polished, wide driver support
Examples	Linux, FreeBSD, Android (AOSP)	Windows, macOS, iOS

Advantages of Open-Source OS	Disadvantages of Open-Source OS
Free of cost and no licence restrictions. Source code can be studied - excellent for learning OS concepts. Customisable for specific needs. Quick bug fixes and strong security through community review. No vendor lock-in; stable and reliable (servers, supercomputers).	Less user-friendly for beginners in some distributions. Limited official support and documentation quality varies. Some hardware drivers and commercial software are not available. Many versions/distributions create confusion; compatibility issues. Source being public may help attackers find vulnerabilities.

Quick Revision Points

- The OS is a resource manager and an interface between user and hardware; the kernel is its core.
- Generations: **1st** vacuum tubes - no OS; **2nd** transistors - batch; **3rd** ICs - multiprogramming, time sharing; **4th** VLSI - PCs, GUI, networks.
- Batch: no interaction, jobs grouped. Multiprogramming: several jobs in memory, CPU utilization ($1 - p^n$). Time sharing: quantum + round robin, quick response.
- Multiprocessor: shared memory and clock, tightly coupled; AMP = master/slave, SMP = peers.
- Distributed: separate memory, loosely coupled, network + message passing, transparency.
- Clustered: independent nodes + shared storage; asymmetric (hot standby) and symmetric clustering; high availability.
- Real time: deadline is critical; hard (catastrophic if missed) and soft (degradation). Examples: VxWorks, QNX.
- Special-purpose: embedded, multimedia, handheld - limited resources, dedicated functions.

- Open source: source code freely available (GNU GPL); Linux by Linus Torvalds (1991), GNU by Richard Stallman (1983).

Poly Notes Hub

Practice Section A: Multiple Choice Questions (15 Marks)

Choose the correct option for each question. The answer key is given at the end of this section.

Q1. An operating system is best described as

- | | | | |
|--|--|--|--|
| (A) A compiler that translates source code into machine code | (B) A program that acts as an interface between the user and the computer hardware | (C) A set of office application programs | (D) A hardware device that controls memory |
|--|--|--|--|

Q2. The first generation of computers (1945 - 1955) was built using

- | | | | |
|-----------------|-------------------------|----------------|------------------|
| (A) Transistors | (B) Integrated circuits | (C) VLSI chips | (D) Vacuum tubes |
|-----------------|-------------------------|----------------|------------------|

Q3. Multiprogramming and spooling were the main features introduced in the

- | | | | |
|----------------------|----------------------|-----------------------|----------------------|
| (A) Third generation | (B) First generation | (C) Second generation | (D) Fifth generation |
|----------------------|----------------------|-----------------------|----------------------|

Q4. In a batch operating system, jobs having similar requirements are

- | | | | |
|--|--------------------------------------|---|--|
| (A) Executed interactively by the user | (B) Executed within strict deadlines | (C) Grouped together and executed one after another | (D) Distributed across networked computers |
|--|--------------------------------------|---|--|

Q5. The main objective of multiprogramming is to

- | | | | |
|--------------------------------|---|-------------------------------------|--|
| (A) Reduce the number of users | (B) Provide response within fixed deadlines | (C) Remove the need for main memory | (D) Increase CPU utilization by always having a job to execute |
|--------------------------------|---|-------------------------------------|--|

Q6. Which technique gives each user a small time quantum on the CPU in round-robin fashion so that all users can interact with their programs?

- | | | | |
|----------------------|------------------|--------------|----------------|
| (A) Batch processing | (B) Time sharing | (C) Spooling | (D) Clustering |
|----------------------|------------------|--------------|----------------|

Q7. In asymmetric multiprocessing,

- | | | | |
|---|---|---|--|
| (A) A master processor controls the system and assigns work to slave processors | (B) All processors are peers and run identical copies of the OS | (C) Every processor has its own memory and OS | (D) Processors are connected through a WAN |
|---|---|---|--|

Q8. Graceful degradation is an advantage of

- | | | | |
|-------------------|---------------------------------|----------------------------|-----------------------------|
| (A) Batch systems | (B) Single-user desktop systems | (C) Multiprocessor systems | (D) Simple handheld systems |
|-------------------|---------------------------------|----------------------------|-----------------------------|

Q9. A distributed system is also called a

- | | | | |
|----------------------------|----------------------------|------------------------|------------------|
| (A) Tightly coupled system | (B) Loosely coupled system | (C) Single-user system | (D) Batch system |
|----------------------------|----------------------------|------------------------|------------------|

Q10. A dedicated system with limited resources designed to perform a specific function, such as in a washing machine, is a(n)

- | | | | |
|----------------------|-------------------------|------------------|---------------------|
| (A) Mainframe system | (B) Time-sharing system | (C) Batch system | (D) Embedded system |
|----------------------|-------------------------|------------------|---------------------|

Q11. In a clustered system, the computers (nodes) generally

- (A) Share a single memory and a single clock (B) Contain only one CPU (C) Are individual systems joined by a fast LAN and sharing storage (D) Never share any storage

Q12. In asymmetric clustering,

- (A) One machine runs the applications while another is in hot-standby mode (B) All machines run applications and monitor each other (C) All machines remain idle (D) Only a single machine exists

Q13. A real-time system in which missing a deadline can cause a catastrophic failure is called a

- (A) Soft real-time system (B) Time-sharing system (C) Batch system (D) Hard real-time system

Q14. Which of the following is an example of a real-time operating system?

- (A) MS-DOS (B) VxWorks (C) IBSYS (D) Windows 3.1

Q15. The Linux kernel was originally developed by

- (A) Richard Stallman (B) Dennis Ritchie (C) Linus Torvalds (D) Bill Gates

Answer Key

Q. No.	Answer	Reason
1	(B)	The OS is an intermediary between user and hardware.
2	(D)	ENIAC / UNIVAC era machines used vacuum tubes.
3	(A)	Third generation (ICs) introduced multiprogramming, spooling and time sharing.
4	(C)	Batch OS runs grouped jobs sequentially without interaction.
5	(D)	Switching to another job when one waits keeps the CPU busy.
6	(B)	Time quantum + round robin is the time-sharing method.
7	(A)	AMP follows a master-slave relationship.
8	(C)	If one processor fails, the others continue (reliability).
9	(B)	Sites have no shared memory/clock and communicate by messages.
10	(D)	Embedded systems are dedicated-function systems.
11	(C)	Cluster = coupled nodes with shared storage, linked by a LAN.
12	(A)	Hot-standby host monitors the active server.
13	(D)	Hard real time: deadline must always be met.
14	(B)	VxWorks is a popular RTOS; others are not real-time.
15	(C)	Linus Torvalds wrote the Linux kernel in 1991.

Practice Section B: Broad Questions (10 Questions)

1. Define an operating system. Explain the goals and the main functions of an operating system with a neat diagram.
2. Describe the evolution of operating systems. Explain the four generations with their hardware technology, features and examples.
3. What is a batch operating system? Explain its working, advantages and disadvantages. What is the role of spooling?
4. Explain multiprogramming, multitasking and time-sharing systems. Differentiate among them and derive the CPU utilization formula for multiprogramming.
5. What are desktop systems? Explain how desktop operating systems have changed from MS-DOS to modern systems, and state their goals.
6. What is a multiprocessor system? Explain its advantages and the types of multiprocessing (asymmetric and symmetric) with a diagram.
7. Define a distributed system. Explain the reasons for building it, its types, and differentiate between network OS and distributed OS.
8. What is a clustered system? Explain asymmetric and symmetric clustering and compare clustered, distributed and multiprocessor systems.
9. What is a real-time system? Explain hard and soft real-time systems with examples, and list the features of a real-time operating system.
10. Write short notes on (a) special-purpose systems: embedded, multimedia and handheld, and (b) open-source operating systems with their advantages, disadvantages and examples.

- End of Chapter 1 -