Operating Systems Notes

May 31, 2025

Contents

1.1 Introduction to Operating Systems

1 Purpose and Core Responsibilities

- **Intermediary:** OS sits between user & applications and hardware, hiding details while exposing services.
- **Goals:**
  - Run programs conveniently (consistent UI).
  - Run programs efficiently (performance, safe sharing).
  - Manage hardware resources (CPU, memory, I/O, storage).

2 Abstraction Layers (high → low)

1. **User** – GUI, CLI, touch, voice.
2. **Application programs** – browsers, compilers, games.
3. **Operating system** – kernel, system programs, middleware.
4. **Computer hardware** – CPU, memory, devices.

3 User and Kernel Mode Transition

Hardware mode bit switches CPU from *user* to *kernel* on a system call (trap) or interrupt; return-from-trap restores user mode.

4 User View vs. System View

A von Neumann CPU follows a fetch–decode–execute loop, issuing a stream of memory addresses that the OS must service quickly and safely.

- **User view:** Focus on ease of use; resource details hidden.
- **System view:**
  - **Resource allocator** – arbitrates CPU, memory, I/O, etc.
  - **Control program** – governs execution, manages devices.

5 What Forms an Operating System?

Always	Usually	Optional / Varies
Kernel – always runs after boot.	**Device drivers**, loadable modules.	**Middleware** (graphics, DB), extra utilities.
	System programs (shells, daemons).	

- **Firmware OSs:** microcode, BIOS/UEFI, and many device controllers contain their own minimal operating systems that run continuously beneath the main kernel.

6 Why Study Operating Systems?

- All code runs on an OS; knowing its policies & APIs yields safer, faster, portable software.
- **OS flavours:** desktop (Windows, macOS), server (Linux, Windows Server), mobile (Android, iOS), and embedded/real-time systems each emphasise different goals.
- **Open-source examples:** Linux, FreeBSD, and Minix show how textbook concepts map to production kernels.
- OS concepts (processes, memory, I/O, security) recur in servers, clouds, IoT.

7 Glossary Highlights (1.1)

Term	Definition
Operating system	Intermediary between user/applications and hardware; hides details, exposes services; runs programs conveniently and efficiently; manages hardware resources.
Kernel	Always runs after boot; provides services.
System program	Loaded at boot time, becomes system daemon; provides services outside the kernel.
Middleware	Optional/varies; graphics, DB, extra utilities.
Resource allocator	Part of OS (system view); arbitrates CPU, memory, I/O, etc.
Control program	Part of OS (system view); governs execution, manages devices.
Ease of use	User view goal; focus on convenience, resource details hidden.
Resource utilization	System view goal; focus on efficiency, performance, safe sharing.

1.2 Computer-System Organization

1 System-Level Hardware Layout

- Shared **system bus** connects CPU cores, main memory, device controllers.
- **Device controller:** logic per device class (disk, GPU, USB) with registers & buffer.
- **Device driver:** kernel API shielding hardware details.
- **Parallelism:** CPU & controllers run concurrently; **memory controller** arbitrates.

2 Interrupts – Hardware-Driven Events

2.1 Lifecycle

1. Controller raises signal on **interrupt-request line**.
2. CPU catches signal, saves state, jumps to handler via **interrupt vector**.
3. Handler services device, restores state, executes `return_from_interrupt`.

2.2 Efficiency Features

- **O(1) dispatch** via vector table.
- **Nonmaskable** vs. **maskable** lines (maskable can be disabled).
- **Priority & interrupt chaining:** high-priority pre-empts low; vector entry may head list of handlers.

2.3 End-to-End I/O Timeline

`User code → I/O request → controller/DMA busy → finish → interrupt → handler → resume user code`

3 Storage Structure

3.1 Memory Hierarchy

Layer	Volatile?	Size	Access	Notes
Registers	Yes	bytes	sub-ns	in CPU
Cache	Yes	KB–MB	ns	SRAM
Main memory	Yes	GBs	∼10 ns	DRAM
NVM/SSD	No	10 GB–TB	μs	electrical
HDD	No	100 GB–TB	ms	mechanical
Optical/tape	No	TB–PB	s–min	archival

3.2 Key Units & Terms

- **Bit** → **byte (8 bits)** → **word** (CPU width).
- **KiB / MiB / GiB / TiB / PiB** = powers of 1024.
- **Volatile** memory loses data on power-off; **non-volatile storage (NVS)** persists.
- NVS divides into *mechanical* media (HDD, optical, tape) and *electrical* non-volatile memory (NVM: SSD, flash).
- **Firmware / EEPROM** stores bootstrap program.
- **Interrupt masking:** the kernel may briefly disable maskable interrupts while executing critical sections to maintain data coherence.

3.3 Why Secondary Storage?

- Main memory is finite and volatile; programs & data live on slower persistent media until loaded.

4 I/O Structure

4.1 Direct Memory Access (DMA)

- Driver configures DMA engine; controller moves block without CPU, raises one interrupt on completion.

4.2 Bus vs. Switched Fabrics

- **Shared bus:** one transfer at a time; common on PCs.
- **Switch fabric:** concurrent links; used in servers & SoCs.

4.3 Full I/O Cycle

1. Driver starts I/O.
2. Controller activates device/DMA.
3. CPU handles other tasks, checks interrupts.
4. Device finishes, raises interrupt.
5. Handler validates data, wakes waiting process.

5 Glossary Highlights (1.2)

Term	Definition
Bus	Shared path linking CPU, memory, controllers.
Device driver	Kernel interface to a controller.
Interrupt / vector / request line	Hardware signal and its dispatch mechanism.
Maskable / nonmaskable interrupt	Can or cannot be disabled.
DMA	Controller-driven block transfer to/from memory.
Volatile / non-volatile memory	Data lost or retained on power loss.

1.3 Computer-system architecture

- Computer systems are organized based on the number of general-purpose processors.

1 Single-processor systems

- **Definition:** One general-purpose CPU with a single processing core.
- Historically common, but few contemporary systems fit this definition.
- **CPU Core:** Executes instructions, contains registers for local data.
- Main CPU core executes general-purpose instruction sets (including processes).
- May include **special-purpose processors** (e.g., disk, keyboard, graphics controllers).
  - Run limited instruction sets; do not run processes.
  - Can be OS-managed (OS sends tasks, monitors status).
  - Example: Disk-controller microprocessor handles disk queue/scheduling, reducing main CPU overhead.
  - Example: Keyboard microprocessor converts keystrokes to codes for CPU.
  - Can be low-level hardware components, operating autonomously without OS communication.
- Use of special-purpose microprocessors does not classify a system as multiprocessor.

2 Multiprocessor systems

- Dominant in modern computing (mobile devices to servers).
- Traditionally: Two or more processors, each with a single-core CPU.
- Share computer bus, sometimes clock, memory, and peripheral devices.
- **Primary advantage:** Increased throughput (more work in less time).
- Speed-up ratio with N processors is less than N due to overhead and contention for shared resources.
- **Symmetric Multiprocessing (SMP):**
  - Each peer CPU processor performs all tasks (OS functions, user processes).
  - Each CPU has its own registers and private/local cache.
  - All processors share physical memory via the system bus.
  - *Figure 1.3.1: Symmetric multiprocessing architecture. Diagram shows main memory connected to two processors, each with its own CPU containing registers and cache.*
  - Benefit: N processes can run simultaneously on N CPUs without significant performance degradation.
  - Inefficiencies from idle/overloaded CPUs are mitigated by sharing data structures.
  - Allows dynamic sharing of processes and resources (e.g., memory) among processors, reducing workload variance.
  - Requires careful design (e.g., CPU Scheduling, Synchronization Tools).
- **Multicore systems:**
  - Modern evolution: Multiple computing cores reside on a single chip.
  - More efficient than multiple single-core chips due to faster on-chip communication.
  - Uses significantly less power than multiple single-core chips (critical for mobile/laptops).
  - Example: Dual-core design (Figure 1.3.2)
    - Each core has its own register set and L1 cache.
    - Shared L2 cache is local to the chip.
    - Combines local (smaller, faster) and shared (higher-level) caches.
    - *Figure 1.3.2: A dual-core design with two cores on the same chip. Diagram shows main memory connected to L2 cache of processor. Processor has two CPU cores, each core has its own registers and L1 cache and shares L2 cache of processor.*
    - An N-core processor appears as N standard CPUs to the OS.
    - Demands efficient use of processing cores from OS/application designers (Threads & Concurrency).
    - Supported by virtually all modern OS (Windows, macOS, Linux, Android, iOS).

Definitions of computer system components

- **CPU:** Hardware that executes instructions.
- **Processor:** A physical chip containing one or more CPUs.
- **Core:** Basic computation unit of the CPU.
- **Multicore:** Multiple computing cores on the same CPU.
- **Multiprocessor:** Multiple processors within the same CPU chip or system.

- General term *CPU* refers to a single computational unit.
- *Core* and *multicore* refer specifically to one or more cores on a CPU.

Non-Uniform Memory Access (NUMA)

- An alternative to scaling multiprocessor systems when bus contention becomes a bottleneck.
- Each CPU (or group of CPUs) has its own **local memory** accessed via a small, fast local bus.
- CPUs are connected by a **shared system interconnect**; all CPUs share one physical address space.
- *Figure 1.3.3: NUMA multiprocessing architecture. Diagram shows NUMA architecture which includes four interconnected CPUs, each attached with its own local memory.*
- **Advantage:** Fast local memory access with no contention over the system interconnect.
- **Result:** NUMA systems scale more effectively with additional processors.
- **Potential drawback:** Increased latency for remote memory access across the system interconnect (e.g., $CPU0$ accessing $CPU3$'s local memory is slower).
- OS minimizes NUMA penalty through careful CPU scheduling and memory management.
- Increasingly popular on servers and high-performance computing systems due to scalability.

Blade servers

- Multiple processor boards, I/O boards, and networking boards placed in the same chassis.
- Differ from traditional multiprocessor systems: each blade-processor board boots independently and runs its own OS.
- Can consist of multiple independent multiprocessor systems.

3 Clustered systems

- Another type of multiprocessor system, gathering multiple CPUs.
- Composed of two or more individual systems (**nodes**) joined together; each node is typically a multicore system.
- Considered **loosely coupled**.
- Definition: Clustered computers share storage and are closely linked via LAN or faster interconnect (e.g., InfiniBand).
- **Primary use:** Provide **high-availability service** (service continues even if one or more systems fail).
  - Achieved by adding redundancy.
  - Cluster software runs on nodes; each node monitors others.
  - If a monitored machine fails, the monitoring machine takes ownership of its storage and restarts applications.
  - Users/clients experience only a brief interruption.
- High availability increases reliability.
- **Graceful degradation:** System's ability to continue providing service proportional to surviving hardware.
- **Fault-tolerant systems:** Can suffer a single component failure and continue operation (requires detection, diagnosis, correction).
- **Clustering structures:**
  - **Asymmetric clustering:** One machine in **hot-standby mode** (monitors active server); if active fails, hot-standby becomes active.
  - **Symmetric clustering:** Two or more hosts run applications and monitor each other; more efficient as it uses all hardware, but requires multiple applications.
- **PC motherboard example:**
  - A fully functioning computer once slots are populated (processor socket, DRAM sockets, PCIe bus slots, I/O connectors).
  - Lowest-cost general-purpose CPUs have multiple cores.
  - Some motherboards have multiple processor sockets (for NUMA systems).
  - *Image shows a computer processor with arrows pointing to: DRAM slots, processor socket, PCI bus slots, and various I/O and power connectors.*
- **High-Performance Computing (HPC) environments:**
  - Clusters provide significantly greater computational power than single-processor/SMP systems.
  - Run applications concurrently on all computers in the cluster.
  - Requires applications to be specifically written for clusters (e.g., **parallelization**).
  - **Parallelization:** Divides a program into separate components that run in parallel on individual cores/computers.
  - Results from nodes are combined into a final solution.
- **Other cluster forms:**
  - **Parallel clusters:** Multiple hosts access the same data on shared storage.
  - Often requires special software/application releases (e.g., Oracle Real Application Cluster).

  - Uses **distributed lock manager (DLM)** for access control and locking.
  - Clustering over Wide-Area Network (WAN).
- Cluster technology is rapidly evolving (supporting thousands of systems, separated by miles).
- Enabled by **Storage-Area Networks (SANs)**: allow many systems to attach to a pool of storage.
- If applications/data are on the SAN, cluster software can assign the application to any host attached to the SAN.
- *Figure 1.3.4: General structure of a clustered system. Diagram shows structure of clustered system in which storage-area network is connected with set of interconnected computers.*

4 Section glossary

Term	**Definition**
Core	Within a CPU, the component that executes instructions.
Multiprocessor systems	Systems with two or more hardware processors (CPU cores) in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.
Symmetric multiprocessing (SMP)	Multiprocessing where each processor performs all tasks, including OS tasks and user processes.
Multicore	Multiple processing cores within the same CPU chip or within a single system.
Multiprocessor	Multiple processors within the same CPU chip or within a single system.
Shared system interconnect	A bus connecting CPUs to memory such that all CPUs can access all system memory; basis for NUMA systems.
Non-Uniform Memory Access (NUMA)	Architecture where memory access time varies based on which core the thread is running on (e.g., core interlink slower than direct DIMM access).
Blade server	Computer with multiple processor, I/O, and networking boards in same chassis; each board boots independently and runs its own OS.
Clustered system	Gathers multiple CPUs; composed of two or more individual systems (nodes) joined together.
High-availability	Service continues even if one or more systems in the cluster fail.
Graceful degradation	System's ability to continue providing service proportional to surviving hardware.
Fault-tolerant system	System that can suffer a single component failure and still continue operation.
Asymmetric clustering	One machine in hot-standby mode monitors active server; takes over if active fails.
Hot-standby mode	Computer in cluster monitors active server, becomes active if it fails.
Symmetric clustering	Two or more hosts run applications and monitor each other.
High-Performance Computing (HPC)	Computing facility for large number of resources used by software designed for parallel operation.
Parallelization	Dividing a program into separate components that run in parallel on individual cores/computers.
Distributed lock manager (DLM)	Function used by clustered system for access control and locking to prevent conflicting operations.
Storage-Area Network (SAN)	Local-area storage network allowing multiple computers to connect to one or more storage devices.

1.4 Operating-system operations

- Provides the environment within which programs are executed.
- Internally, OS vary greatly, but share commonalities.

1 Initial Program and Kernel Loading

- Computer needs an initial program to run when powered up or rebooted.
- This **bootstrap program** is typically simple and stored in **firmware**.
- Initializes all system aspects: CPU registers, device controllers, memory contents.
- Must locate and load the **operating-system kernel** into memory.
- Once kernel is loaded and executing, it provides services.
- Some services are provided by **system programs** loaded at boot time, becoming **system daemons**.
- Example: On Linux, `systemd` starts many other daemons.
- After booting, the system waits for events (e.g., interrupts or traps).

Hadoop

- Open-source software framework for distributed processing of **big data** in clustered systems.
- Designed to scale from a single system to thousands of computing nodes.
- Assigns tasks to nodes, manages communication for parallel computations, and coalesces results.
- Detects and manages node failures, providing efficient and reliable distributed computing.
- Organized around three components:
  1. A distributed file system for managing data and files across nodes.
  2. The **YARN** ("Yet Another Resource Negotiator") framework for resource management and task scheduling.
  3. The **MapReduce** system for parallel data processing across nodes.
- Primarily runs on Linux systems.
- Applications can be written in PHP, Perl, Python, and Java (popular due to Java libraries for MapReduce).

2 Interrupts and Traps

- Events are almost always signaled by the occurrence of an **interrupt**.
- A **trap** (or an **exception**) is a software-generated interrupt.
  - Caused by an error (e.g., division by zero, invalid memory access).
  - Caused by a specific request from a user program for an operating-system service, by executing a **system call**.

3 Multiprogramming and Multitasking

- Important aspects of OS: ability to run multiple programs.
- Single program cannot keep CPU or I/O devices busy at all times.
- **Multiprogramming:**
  - Increases CPU utilization and user satisfaction.
  - OS keeps several **processes** (programs in execution) in memory simultaneously.
  - When a process waits for a task (e.g., I/O), OS switches to another process.
  - CPU is never idle as long as at least one process needs to execute.
  - *Figure 1.4.1: Memory layout for a multiprogramming system. Diagram shows vertical column representing memory which contains layers of operating system, process 1, process 2, process 3, and process 4.*
- **Multitasking:**
  - Logical extension of multiprogramming.
  - CPU switches frequently among processes, providing fast **response time**.
  - Handles slow interactive I/O (e.g., user typing speed).
  - Requires:
    * Memory management (*Main Memory*, *Virtual Memory* chapters).
    * **CPU scheduling** (choosing which process runs next).
    * Protection mechanisms to limit processes' ability to affect one another.
    * **Virtual memory**: allows execution of processes not completely in memory; enables running programs larger than physical memory; abstracts main memory into a large, uniform array of storage, separating **logical memory** from physical memory.
    * File system (*File-System Interface*, *File-System Implementation*, *File-System Internals* chapters).

    - Storage management (*Mass-Storage Structure* chapter).
    - Resource protection (*Protection* chapter).
    - Process synchronization and communication (*Synchronization Tools*, *Synchronization Examples* chapters).
    - Deadlock prevention (*Deadlocks* chapter).

4 Dual-mode and Multimode Operation

- Ensures incorrect/malicious programs cannot cause other programs or the OS to execute incorrectly.
- Distinguishes between execution of operating-system code and user-defined code.
- Hardware support provides differentiation among various modes of execution.
- At least two separate modes of operation:
  - **User mode**: System executing on behalf of a user application.
  - **Kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**): System executing on behalf of the operating system.
- A **mode bit** is added to hardware to indicate current mode: kernel (0) or user (1).
- *Figure 1.4.2: Transition from user to kernel mode. Diagram shows events such as user process executing, calling system call and return from system call in user mode whereas trapping mode bit 0, execute system call, and returning mode bit 1 in kernel mode.*
- At system boot time, hardware starts in kernel mode.
- OS is loaded and starts user applications in user mode.
- Whenever a trap or interrupt occurs, hardware switches from user mode to kernel mode (mode bit to 0).
- OS always gains control in kernel mode.
- System always switches to user mode (mode bit to 1) before passing control to a user program.
- *Figure 1.4.3: Windows Performance Monitor tool. Image shows a graph which utilizes an x-axis with time and y-axis with percentages of system usage. It shows in green the user mode which is higher than the red line which is in kernel mode.*
- Dual mode provides protection for OS from errant users and users from one another.
- **Privileged instructions**: Machine instructions that may cause harm, executable only in kernel mode.
  - Attempt to execute in user mode → hardware treats as illegal and traps to OS.
  - Examples: switch to kernel mode, I/O control, timer management, interrupt management.
- Modes can be extended beyond two:
  - Intel processors: four separate **protection rings** (ring 0 kernel, ring 3 user).
  - ARM v8 systems: seven modes.
  - CPUs supporting virtualization: separate mode for **Virtual Machine Manager (VMM)**.
- Life cycle of instruction execution: OS (kernel mode) → user application (user mode) → back to OS via interrupt, trap, or system call.
- Most contemporary OS (Microsoft Windows, Unix, Linux) use dual-mode for protection.

System Calls

- Provide means for a user program to ask the OS to perform tasks reserved for the OS.
- Invoked via a trap to a specific location in the interrupt vector.
- Can be executed by a generic 'trap' instruction or a specific 'syscall' instruction.
- Treated by hardware as a software interrupt.
- Control passes to a service routine in the OS, and mode bit is set to kernel mode.
- Kernel examines interrupting instruction, determines system call, verifies parameters, executes request, and returns control.

Program Errors

- Hardware protection detects errors that violate modes (e.g., illegal instruction, accessing memory not in user's address space).
- Hardware traps to the OS.
- OS terminates the program abnormally, provides an error message, and may dump program memory to a file for examination/correction.

5 Timer

- Ensures OS maintains control over the CPU.
- Prevents user programs from getting stuck in infinite loops or failing to return control to the OS.
- Can be set to interrupt the computer after a specified period (fixed or variable).
- Variable timer implemented by a fixed-rate clock and a counter.
  - OS sets the counter.
  - Every clock tick, counter is decremented.
  - When counter reaches 0, an interrupt occurs.
- OS ensures timer is set before turning control to user.
- If timer interrupts, control transfers automatically to OS (may treat as fatal error or give more time).
- Instructions that modify the content of the timer are privileged.

Linux timers

- Kernel configuration parameter **HZ** specifies frequency of timer interrupts (e.g., 250 HZ = 250 interrupts/sec).
- Related kernel variable **jiffies** represents number of timer interrupts since system boot.

6 Section glossary

Term	**Definition**
Big data	Extremely large sets of data; distributed systems are well suited to working with big data.
MapReduce	Google-created big data programming model and implementation for parallel processing across nodes in a distributed cluster.
System daemon	Service provided outside the kernel by system programs loaded at boot time and running continuously.
Trap	Software interrupt caused by an error or a specific request from a user program for an operating-system service.
Exception	Software-generated interrupt caused by an error or a specific request from a user program for an operating-system service.
System call	Software-triggered interrupt allowing a process to request a kernel service.
Multiprogramming	Technique that increases CPU utilization by organizing jobs so that the CPU always has a job to execute.
Process	A program loaded into memory and executing.
Multitasking	Concurrent performance of multiple jobs; CPU switches frequently among them for fast response time.
Response time	Amount of time it takes the system to respond to user action.
CPU scheduling	Process by which the system chooses which job will run next if several jobs are ready to run.
Virtual memory	Technique that allows execution of a process not completely in memory; also, separation of computer memory address space from physical into logical.
Logical memory	Memory as viewed by the user; usually a large uniform array, not matching physical memory in virtual memory systems.
User mode	CPU mode for executing user processes in which some instructions are limited or not allowed.
Kernel mode	CPU mode in which all instructions are enabled; the kernel runs in this mode.
Supervisor mode	CPU mode in which all instructions are enabled; the kernel runs in this mode.
System mode	CPU mode in which all instructions are enabled; the kernel runs in this mode.

1.5 Resource management

- An operating system is a **resource manager**, managing CPU, memory space, file-storage space, and I/O devices.

1 Process management

- A program in execution is a **process** (e.g., compiler, word-processing program, social media app).
- A process needs resources: CPU time, memory, files, and I/O devices, allocated during its execution.
- Initialization data (input) may be passed to a process (e.g., URL for a web browser).
- OS reclaims reusable resources when a process terminates.
- A program is a *passive* entity (like a file on disk), whereas a process is an *active* entity.
- A **single-threaded process** has one **program counter** specifying the next instruction. Execution is sequential.
- A **multithreaded process** has multiple program counters. (Covered in *Threads & Concurrency*).
- A process is the unit of work in a system.
- A system consists of operating-system processes (execute system code) and user processes (execute user code).
- Processes can execute concurrently (multiplexing on a single CPU core) or in parallel (across multiple CPU cores).
- The OS is responsible for:
  - Creating and deleting both user and system processes.
  - Scheduling processes and threads on the CPUs.
  - Suspending and resuming processes.
  - Providing mechanisms for process synchronization.
  - Providing mechanisms for process communication.
- (Process management techniques discussed in chapters *Processes* through *Synchronization Examples*).

2 Memory management

- Main memory is central to modern computer operation: a large array of bytes, each with an address.
- It's a repository of quickly accessible data shared by the CPU and I/O devices.
- CPU reads instructions and data from main memory (von Neumann architecture).
- Main memory is generally the only large storage CPU accesses directly. Data from disk must be transferred to main memory.
- For a program to execute, it must be mapped to absolute addresses and loaded into memory.
- To improve CPU utilization and system responsiveness, general-purpose computers keep several programs in memory.
- Memory-management schemes vary; effectiveness depends on the situation and *hardware design*.
- The OS is responsible for:
  - Keeping track of which parts of memory are currently being used and by which process.
  - Allocating and deallocating memory space as needed.
  - Deciding which processes (or parts of processes) and data to move into and out of memory.
- (Memory-management techniques discussed in chapters *Main Memory* and *Virtual Memory*).

3 File-system management

- OS provides a uniform, logical view of information storage.
- OS abstracts from physical properties of storage devices to define a logical storage unit: the **file**.
- OS maps files onto physical media and accesses them via storage devices.
- Computers store information on various physical media (secondary, tertiary).
- A **file** is a collection of related information defined by its creator (e.g., programs, data).
- Files can be free-form (text files) or rigidly formatted (fixed fields, e.g., mp3).
- OS implements the file concept by managing mass storage media and their control devices.
- Files are normally organized into directories.
- Access control may be needed for multiple users (read, write, append).
- The OS is responsible for:
  - Creating and deleting files.
  - Creating and deleting directories to organize files.
  - Supporting primitives for manipulating files and directories.
  - Mapping files onto mass storage.
  - Backing up files on stable (nonvolatile) storage media.

- (File-management techniques discussed in chapters *File-System Interface*, *File-System Implementation*, and *File-System Internals*).

4 Mass-storage management

- Computer systems use secondary storage (HDDs, NVM devices) to back up main memory.
- Most programs are stored on these devices until loaded into memory.
- Proper management of secondary storage is crucial.
- The OS is responsible for secondary storage management:
  - Mounting and unmounting.
  - Free-space management.
  - Storage allocation.
  - Disk scheduling.
  - Partitioning.
  - Protection.
- Tertiary storage (slower, lower cost, higher capacity) is used for backups, seldom-used data, archival.
  - Examples: magnetic tape, CD, DVD, Blu-ray.
  - OS may manage mounting/unmounting, device allocation, data migration.
- (Techniques discussed in chapter *Mass-Storage Structure*).

5 Cache management

- **Caching** is an important principle: information is copied from a storage system (e.g., main memory) to a faster temporary storage system (the **cache**).
- When information is needed, check cache first. If not present, use source and copy to cache.
- Internal programmable registers act as a high-speed cache for main memory.
- Hardware-only caches (e.g., instruction cache, data caches) are outside OS control.
- **Cache management** is important due to limited cache size; careful selection of size and replacement policy increases performance.
- *Figure 1.5.1: Characteristics of various types of storage. Table lists level, typical size, implementation technology, access time, bandwidth, managed by, and backed by for various types of storage such as registers, cache, main memory, solid-state disk, and magnetic disk.*
- Information movement between storage levels can be explicit (OS-controlled, e.g., disk to memory) or implicit (hardware function, e.g., cache to CPU).
- Data coherency:
  - The same data may appear in different levels of the storage hierarchy (e.g., integer A on disk, main memory, cache, internal register).
  - *Figure 1.5.2: Migration of integer A from disk to register. Block diagram shows migration of integer A from magnetic disk to hardware register through main memory and cache.*
  - In a multitasking environment, processes must access the most recently updated value.
  - In a multiprocessor environment, an update to a value in one cache must be reflected in all other caches where it resides. This is **cache coherency** (usually a hardware issue).
  - In a distributed environment, replicas of the same file on different computers must be kept consistent. (Discussed in *Networks and Distributed Systems*).
- (Replacement algorithms for software-controlled caches discussed in chapter *Virtual Memory*).

6 I/O system management

- OS hides peculiarities of specific hardware devices from the user.
- The **I/O subsystem** consists of:
  - A memory-management component (buffering, caching, spooling).
  - A general device-driver interface.
  - Drivers for specific hardware devices.
- Only the device driver knows the peculiarities of its specific device.
- (Discussed in chapter *I/O Systems*).

7 Section glossary

Term	Definition
Resource manager	The role of an operating system in managing the computer's resources.
Program counter	A CPU register indicating the main memory location of the next instruction to load and execute.
File	The smallest logical storage unit; a collection of related information defined by its creator.
Caching	The use of temporary data storage areas to improve performance.
Cache management	The management of a cache's contents.
Cache coherency	The coordination of the contents of caches such that an update to a value stored in one cache is immediately reflected in all other caches that hold that value.
I/O subsystem	The I/O devices and the part of the kernel that manages I/O.

1.6 Security and protection

- In multi-user, multi-process systems, access to data must be regulated.
- Mechanisms ensure processes operate only on authorized resources (files, memory segments, CPU, etc.).
- Examples: memory-addressing hardware (process within its address space), timer (CPU relinquishing control), device-control registers (not user accessible).

1 Protection

- **Definition:** Any mechanism for controlling the access of processes or users to the resources defined by a computer system.
- Must provide means to specify and enforce controls.
- Improves **reliability** by detecting latent errors at component subsystem interfaces.
- Prevents contamination of healthy subsystems by malfunctioning ones.
- An unprotected resource cannot defend against unauthorized or incompetent use.
- Distinguishes between authorized and unauthorized usage (discussed further in chapter *Protection*).

2 Security

- **Definition:** The defense of a system from external and internal attacks.
- Attacks include: viruses, worms, **denial-of-service** (DoS) attacks, identity theft, and theft of service.
- Prevention can be an operating-system function, policy, or require additional software.
- Operating-system security features are a fast-growing area of research and implementation due to rising incidents (discussed further in chapter *Security*).

3 User and Group Identification

- Systems must distinguish among all users.
- Most operating systems maintain a list of user names and associated **user identifiers** (**user IDs** or **UIDs**).
- In Windows, this is a **security ID** (**SID**).
- These numerical IDs are unique per user.
- Upon login, authentication determines the user ID, which is then associated with all of the user's processes and threads.
- For readability, user IDs are translated back to user names.
- To distinguish among sets of users (e.g., file owner vs. a group of readers), systems define a **group name** and the set of users belonging to that group.
- Group functionality is implemented as a system-wide list of group names and **group identifiers**.
- A user can belong to one or more groups, depending on OS design.
- Group IDs are also included in every associated process and thread.

4 Privilege Escalation

- Users sometimes need to **escalate privileges** to gain extra permissions for an activity (e.g., access to a restricted device).
- Operating systems provide various methods for this.
- On UNIX, the **setuid** attribute on a program causes it to run with the user ID of the file's owner, rather than the current user's ID.
- The process runs with this **effective UID** until it turns off the extra privileges or terminates.

5 Section glossary

Term	**Definition**
Protection	Any mechanism for controlling the access of processes or users to the resources defined by a computer system.
Security	The defense of a system from external and internal attacks (e.g., viruses, DoS, identity theft).
User identifier (user ID) (UID)	A unique numerical user identifier.
Security ID (SID)	In Windows, a value used to uniquely identify a user or group for security purposes.
Group identifier	Similar to a user identifier, but used to identify a group of users to determine access rights.
Escalate privileges	To gain extra permissions for an activity, as when a user needs access to a restricted device.
Setuid	A UNIX attribute on a program that causes it to run with the user ID of the file's owner.
Effective UID	The UID the process is currently using, which can be different from the login UID due to privilege escalation.

1.7 Virtualization

- **Definition:** Technology that abstracts a single computer's hardware (CPU, memory, disk, network) into several execution environments.
- Creates the illusion that each environment runs on its own private computer.
- Environments can be different OS (e.g., Windows, UNIX) running concurrently and interacting.
- User can switch among virtual machines (VMs) like switching among processes in a single OS.
- Allows OS to run as applications within other OS.
- Industry is vast and growing due to its utility.

1 Emulation vs. Virtualization

Emulation

- Simulates computer hardware in software.
- Used when **source CPU type** differs from **target CPU type**.
- **Example:** Apple's "Rosetta 2" allowed Intel x86 applications to run on ARM-based Apple Silicon.
- **Drawback:** Heavy performance cost; every machine-level instruction translated, often resulting in multiple target instructions.
- Emulated code runs much slower than native code.

Virtualization

- An OS natively compiled for a specific CPU architecture runs within another OS also native to that CPU.
- **Origin:** First used on IBM mainframes for multiple users to run tasks concurrently on a single-user system.
- **VMware's innovation:** Created virtualization technology as an application running on Windows for Intel x86 CPUs.
  - Ran guest copies of Windows or other native x86 OS.
  - Windows acted as the **host operating system**.
  - VMware application was the **virtual machine manager (VMM)**.
  - **VMM** responsibilities: runs guest OS, manages resources, protects guests from each other.
- *Figure 1.7.1: A computer running (a) a single operating system and (b) three virtual machines. Diagram shows a single OS system with hardware, kernel, programming interface, and one process. In contrast, a virtualized system shows hardware, a virtual machine manager, three virtual machines, each with its own kernel, programming interface, and processes.*

2 Growing Importance of Virtualization

- Despite modern OS running multiple applications reliably, virtualization use continues to grow.
- **On laptops/desktops:**
  - Allows users to install multiple OS for exploration.
  - Run applications for OS other than the native host (e.g., macOS x86 running Windows 10 guest).
- **For software development companies:**
  - Run all target OS on a single physical server for development, testing, and debugging.
- **Within data centers:**
  - Common method for executing and managing computing environments.
  - Modern VMMs (e.g., VMware ESX, Citrix XenServer) *are* the host OS, providing services and resource management directly to VM processes.
- **Educational use:** Linux VM provided with this text allows running Linux and development tools regardless of host OS.

3 Section glossary

Term	**Definition**
Virtualization	A technology for abstracting the hardware of a single computer into several different execution environments, creating the illusion that each environment runs on its own private computer.
Virtual machine (VM)	The abstraction of hardware allowing a virtual computer to execute on a physical computer; multiple VMs can run on a single physical machine, each with a different OS.
Emulation	A methodology to enable a process to run when the compiled program's original (source) CPU type differs from the target CPU type.
Guest	In virtualization, an operating system running in a virtual environment rather than natively on the computer hardware.
Host	In virtualization, the location of the virtual machine manager, which runs guest operating systems; generally, a computer.
Virtual machine manager (VMM)	The computer function that manages the virtual machine; also called a **hypervisor**.

1.8 Distributed systems

- Collection of physically separate, possibly heterogeneous computer systems networked together.
- Provides users access to various resources maintained by the system.
- Increases computation speed, functionality, data availability, and reliability.
- OS may generalize network access as file access (details in device driver) or require specific network function invocation.
- Protocols greatly affect utility and popularity.

1 Networks

- **Definition:** A communication path between two or more systems.
- Distributed systems depend on networking.
- Vary by protocols, distances between nodes, and transport media.
- **TCP/IP:** Most common network protocol, fundamental architecture of the Internet.
  - Supported by most general-purpose OS.
  - Some systems use proprietary protocols.
- OS requires a network protocol to have an interface device (e.g., network adapter) with a device driver and software for data handling.

Network Characterization by Distance

- **Local-area network (LAN):** Connects computers within a room, building, or campus.
- **Wide-area network (WAN):** Links buildings, cities, or countries (e.g., global company offices).
- **Metropolitan-area network (MAN):** Links buildings within a city.
- **Personal-area network (PAN):** Wireless technology (e.g., Bluetooth, 802.11) for communication over several feet (e.g., phone to headset).

Network Media

- Varied media: copper wires, fiber strands, wireless transmissions (satellites, microwave dishes, radios).
- Cellular phone connections create networks.
- Short-range infrared communication can be used.
- Any communication between computers uses or creates a network.
- Networks vary in performance and reliability.

2 Network Operating Systems vs. Distributed Operating Systems

- Some OS extend beyond basic network connectivity.
- **Network operating system:**
  - Provides features like file sharing across the network.
  - Includes a communication scheme for processes on different computers to exchange messages.
  - Computer runs autonomously but is aware of and communicates with other networked computers.
- **Distributed operating system:**
  - Provides a less autonomous environment.
  - Computers communicate closely enough to create the illusion of a single OS controlling the network.

3 Section glossary

Term	**Definition**
Network	A communication path between two or more systems.
Transmission Control Protocol/Internet Protocol (TCP/IP)	The most common network protocol; provides the fundamental architecture of the Internet.
Local-area network (LAN)	A network that connects computers within a room, a building, or a campus.
Metropolitan-area network (MAN)	A network linking buildings within a city.
Personal-area network (PAN)	A network linking devices within several feet of each other (e.g., on a person).
Wide-area network (WAN)	A network which usually links buildings, cities, or countries.
Network operating system	A type of operating system that provides features such as file sharing across a network, along with a communication scheme that allows different processes on different computers to exchange messages.

1.9 Kernel data structures

- Fundamental data structures are used extensively in operating systems.

1 Lists, stacks, and queues

- **Array:** Simple data structure where each element can be accessed directly (e.g., main memory).
- **List:** Collection of data values as a sequence; items must be accessed in a particular order.
  - Implemented commonly as a **linked list**, where items are linked.
  - **Singly linked list:** Each item points to its successor.
  - *Figure 1.9.1: Singly linked list. Diagram shows a sequence of data sets where each data set points to its immediate successor, and the last data set points to null.*
  - **Doubly linked list:** A given item can refer to its predecessor or successor.
  - *Figure 1.9.2: Doubly linked list. Diagram shows a sequence of data sets where each data set points to its immediate successor and predecessor.*
  - **Circularly linked list:** The last element refers to the first element, not null.
  - *Figure 1.9.3: Circularly linked list. Diagram shows a sequence of data sets where each data set points to its immediate successor, and the last data set points to the first data set instead of null.*
  - Accommodate varying item sizes; allow easy insertion/deletion.
  - **Disadvantage:** Retrieving a specified item is $O(N)$ in worst case (linear performance).
  - Used directly by kernel algorithms or for constructing other structures.
- **Stack:** Sequentially ordered data structure using **Last In, First Out (LIFO)** principle.
  - Last item added is first removed.
  - Operations: **push** (insert), **pop** (remove).
  - OS uses stacks for function calls (parameters, local variables, return address pushed/popped).
- **Queue:** Sequentially ordered data structure using **First In, First Out (FIFO)** principle.
  - Items removed in order of insertion.
  - Common in OS (e.g., printer jobs, CPU scheduling tasks).

2 Trees

- Data structure to represent data hierarchically via parent-child relationships.
- **General tree:** Parent may have unlimited children.
- **Binary tree:** Parent has at most two children (left and right).
- **Binary search tree:** Binary tree with ordering: $left\_child \leq right\_child$.
  - *Figure 1.9.4: Binary search tree. Diagram shows a binary search tree with a root item 17, branching into 12 and 35. Item 12 branches into 6 and 14. Item 35 branches into 40, which then branches into 38.*
  - Worst-case search performance is $O(N)$.
  - **Balanced binary search tree:** Contains N items with at most $\lg N$ levels, ensuring $O(\lg N)$ worst-case performance.
  - **Example:** Linux uses a balanced binary search tree (red-black tree) for CPU scheduling.

3 Hash functions and maps

- **Hash function:** Takes data input, performs numeric operation, returns numeric value.
- Numeric value used as index into a table (array) for quick data retrieval.
- Retrieval can be $O(1)$ (constant time), much faster than $O(N)$ for lists.
- Used extensively in OS due to performance.
- **Hash collision:** Two unique inputs result in the same output value (same table location).
  - Handled by having a linked list at the table location for all items with the same hash value.
  - More collisions reduce efficiency.
- **Hash map:** Associates (maps) [key:value] pairs using a hash function.
  - Apply hash function to key to obtain value.
  - *Figure 1.9.5: Hash map. Diagram shows a hash map with bits numbered 0 to N. A hash function applied to a key on one bit of the hash map obtains a value.*
  - **Example:** User name mapped to password for authentication.

4 Bitmaps

- String of N binary digits representing the status of N items.
- **Example:** 0 = resource available, 1 = resource unavailable.
- Value of i^{th} position associated with i^{th} resource.
- **Example bitmap:** ‘0 0 1 0 1 1 1 0 1‘ (resources 2, 4, 5, 6, 8 unavailable; 0, 1, 3, 7 available).
- **Space efficiency:** Significant power; using a single bit vs. eight-bit Boolean value saves space.
- Commonly used to represent availability of large number of resources (e.g., disk blocks).

5 Linux kernel data structures

- Kernel source code provides details.
- Linked-list: ‘`<linux/list.h>`‘.
- Queue: known as ‘`kfifo`‘, implementation in ‘`kfifo.c`‘.
- Balanced binary search tree: implemented as **red-black trees**, details in ‘`<linux/rbtree.h>`‘.

6 Section glossary

Term	**Definition**
List	A data structure that presents a collection of data values as a sequence.
Linked list	A data structure in which items are linked to one another.
Stack	A sequentially ordered data structure that uses the last-in, first-out (LIFO) principle for adding and removing items.
Queue	A sequentially ordered data structure that uses the first-in, first-out (FIFO) principle.
Tree	A data structure that can be used to represent data hierarchically; data values are linked through parent-child relationships.
General tree	A tree data structure in which a parent may have unlimited children.
Binary tree	A tree data structure in which a parent may have at most two children.
Binary search tree	A type of binary tree data structure that requires an ordering between the parent's two children (left child $\leq$ right child).
Balanced binary search tree	A tree containing N items that has, at most, $\lg N$ levels, ensuring worst-case performance of $O(\lg N)$.
Red-black tree	A tree containing N items and having at most $\lg N$ levels, thus ensuring worst-case performance of $O(\lg N)$.
Hash function	A function that takes data as its input, performs a numeric operation on the data, and returns a numeric value.
Hash map	A data structure that maps [key:value] pairs using a hash function.
Bitmap	A string of N binary digits that can be used to represent the status of N items.

1.10 Computing environments

- Operating systems are used in a variety of computing environments.

1 Traditional computing

- Lines between traditional computing environments have blurred.
- **Typical office environment** evolved from PCs on a network with file/print servers to:
  - **Web technologies** and increasing **WAN bandwidth** stretching boundaries.
  - Companies use **portals** for web accessibility to internal servers.
  - **Network computers** (or **thin clients**) used for security/easier maintenance.
  - Mobile computers synchronize with PCs for portable use.
  - Mobile devices connect to wireless/cellular networks for web portals.
- **Home computing** evolved from single PCs with slow modems to:
  - Inexpensive fast data connections.
  - Home computers serving web pages and running networks (printers, client PCs, servers).
  - Use of **firewalls** to protect networks from security breaches.
- Historically, systems were either **batch** (processed jobs in bulk with predetermined input) or **interactive** (waited for user input).
- **Time-sharing systems:**
  - Multiple users shared time on systems to optimize resource use.
  - Used a timer and scheduling algorithms to cycle processes rapidly through the CPU.
  - Rare today as traditional time-sharing systems.
  - Same scheduling technique used on desktops, laptops, servers, mobile computers.
  - Processes often owned by a single user (or user and OS).
  - User and system processes managed to get a slice of computer time (e.g., multiple windows, web browser processes).

2 Mobile computing

- **Definition:** Computing on handheld smartphones and tablet computers.
- **Physical features:** Portable and lightweight.
- **Functionality evolution:**
  - Historically: sacrificed screen size, memory, functionality for mobile access (email, web browsing).
  - Today: rich features, comparable to laptops, offering unique/impractical functionality for desktops.
  - Used for music, video, digital books, photos, HD video editing.
- **Unique features leveraged by applications:**
  - **GPS chips:** Determine precise location for navigation apps.
  - **Accelerometers and gyroscopes:** Detect orientation, tilting, shaking (e.g., in games).
  - **Augmented-reality applications:** Overlay information on a display of the current environment.
- **Connectivity:** Typically use **IEEE 802.11 wireless** or **cellular data networks**.
- **Limitations compared to PCs:**
  - More limited memory capacity and processing speed (e.g., 256 GB storage vs. 8 TB on desktop).
  - Smaller, slower processors with fewer cores due to power consumption concerns.
- **Dominant mobile operating systems:**
  - **Apple iOS:** Designed for Apple iPhone and iPad.
  - **Google Android:** Powers smartphones and tablets from many manufacturers.

3 Client-server computing

- **Definition:** Network architecture where **server systems** satisfy requests from **client systems**.
- A form of specialized distributed system.
- *Figure 1.10.1: General structure of a client-server system. Diagram shows server connected through network to client desktop, client laptop, and client smartphone.*
- **Server system categories:**
  - **Compute-server system:** Client sends request to perform an action (e.g., read data); server executes and sends results (e.g., database server).
  - **File-server system:** Provides a file-system interface for clients to create, update, read, delete files (e.g., web server delivering files).

4 Peer-to-peer computing

- **Definition:** Distributed system model where clients and servers are not distinguished; all nodes are **peers**.
- Each node can act as either a client (requesting service) or a server (providing service).
- **Advantage:** Services can be provided by several distributed nodes, avoiding server bottlenecks of client-server systems.
- **Node participation:** Must first join the network of peers.
- **Service discovery methods:**
  - **Centralized lookup service:** Node registers service; clients contact lookup service to find providers. Communication then occurs directly between client and provider (e.g., Napster).
  - **No centralized lookup service:** Client broadcasts request to all nodes; providers respond directly. Requires a **discovery protocol** (e.g., Gnutella).
  - *Figure 1.10.2: Peer-to-peer system with no centralized service. Image shows a client broadcasting a request to other nodes, and a service provider node responding directly to the client.*
- **Examples:**
  - **Napster** (late 1990s): Centralized server indexed files, actual exchange was peer-to-peer. Shut down due to copyright infringement.
  - **Gnutella:** Decentralized, clients broadcasted requests.
  - **Skype:** Hybrid approach with centralized login server but decentralized peer communication for VoIP calls and messages.

5 Cloud computing

- **Definition:** Delivers computing, storage, and applications as a service across a network.
- Logical extension of **virtualization**, using it as a base.
- **Example: Amazon Elastic Compute Cloud (EC2)** with thousands of servers, millions of VMs, petabytes of storage. Users pay for resources used. **Types of cloud computing:** (not discrete, often combined)
  - **Public cloud:** Available via Internet to anyone willing to pay.
  - **Private cloud:** Run by a company for its own use.
  - **Hybrid cloud:** Includes both public and private components.
  - **Software as a service (SaaS):** Applications (e.g., word processors) available via Internet.
  - **Platform as a service (PaaS):** Software stack ready for application use via Internet (e.g., database server).
  - **Infrastructure as a service (IaaS):** Servers or storage available over Internet (e.g., backup storage).
- **Underlying infrastructure:**
  - Traditional operating systems within cloud infrastructure.
  - **VMMs** (Virtual Machine Managers) manage virtual machines.
  - **Cloud management tools** (e.g., VMware vCloud Director, Eucalyptus) manage VMMs and cloud resources, acting as a new type of OS.
- *Figure 1.10.3: Cloud computing. Diagram shows a public cloud providing IaaS, with cloud services and user interface protected by a firewall.*

6 Real-time embedded systems

- **Embedded computers:** Most prevalent form of computers (car engines, robots, optical drives, microwaves).
- Tend to have very specific tasks.
- Systems are usually primitive; OS provides limited features, often little/no user interface.
- Focus on monitoring and managing hardware devices.
- **Variations:**
  - General-purpose computers running standard OS (e.g., Linux) with special applications.
  - Hardware devices with special-purpose embedded OS.
  - Hardware devices with **application-specific integrated circuits (ASICs)** that perform tasks without an OS.
- Expanding use: computerized houses controlling heating, lighting, alarms, appliances; potential for smart refrigerators.
- **Real-time operating systems (RTOS):**
  - Used when rigid time requirements are placed on processor operation or data flow.
  - Often used as control devices in dedicated applications (e.g., scientific experiments, medical imaging, industrial control, weapon systems, fuel-injection).
  - Have well-defined, fixed time constraints.
  - Processing *must* be done within defined constraints, or the system fails (e.g., robot arm halting after smashing).
  - Functions correctly only if it returns the correct result within its time constraints.
  - Contrasts with traditional systems where quick response is desirable but not mandatory.

7 Section glossary

Term	Definition
Portals	Gateways between requestors and services running on provider computers.
Network computer	A limited computer that understands only web-based computing.
Thin client	A limited computer (terminal) used for web-based computing.
Wireless network	A communication network composed of radio signals rather than physical wires.
Firewall	A computer, appliance, process, or network router that sits between trusted and untrusted systems or devices; protects a network from security breaches by managing and blocking certain types of communications.
Mobile computing	A mode of computing involving small portable devices like smartphones and tablet computers.
Apple iOS	The mobile operating system created by Apple Inc.
Google Android	The mobile operating system created by Google Inc.
Server system	A system providing services to other computers (e.g., a web server).
Client system	A computer that uses services from other computers (such as a web client).
Client-server model	A mode of computing in which a server provides services to one or more clients.
Compute-server system	A server that provides an interface to which a client can send a request for an action (e.g., read data); server executes and sends results to client.
File-server system	A server that provides a file-system interface where clients can create, update, read, and delete files (e.g., a web server that delivers files to clients running web browsers).
Cloud computing	A type of computing that delivers computing, storage, and even applications ”as a service” across a network.
Amazon Elastic Compute Cloud (EC2)	An instance of cloud computing implemented by Amazon.
Public cloud	Cloud computing available via the Internet to anyone willing to pay for the services offered.
Private cloud	Cloud computing run by a company for that company’s own use.
Hybrid cloud	A type of cloud computing that includes both public and private cloud components.
Software as a Service (SaaS)	A type of computing in which one or more applications (such as word processors or spreadsheets) are available as a service via the Internet.
Platform as a Service (PaaS)	A software stack ready for application use via the Internet (e.g., a database server).
Infrastructure as a Service (IaaS)	A type of computing in which servers or storage are available over the Internet (e.g., storage available for making backup copies of production data).
ASIC	An application-specific integrated circuit (hardware chip) that performs its tasks without an operating system.
Real-time operating systems (RTOS)	Systems used when rigid time requirements have been placed on the operation of a processor or the flow of data; often used as control devices in dedicated applications.

1.11 Free and open-source operating systems

- Study of OS made easier by free software and open-source releases.
- Both available in source-code format (not compiled binary).
- **Free software** (*free/libre software*):
  - Source code available.
  - Licensed for no-cost use, redistribution, and modification.
- **Open-source software**:
  - Source code available.
  - Does not necessarily offer free licensing.
- All free software is open source, but some open-source software is not ”free.”
- **GNU/Linux:** Most famous open-source OS; some distributions are free, others open source only.
- **Microsoft Windows:** Closed-source, proprietary software (Microsoft owns, restricts use, protects source code).
- **Apple macOS:** Hybrid approach (open-source **Darwin** kernel, proprietary closed-source components).
- Starting with source code allows programmers to produce executable binary code.
- Examining source code is an excellent learning tool: students can modify, compile, and run changes.
- **Benefits of open-source operating systems:**
  - Community of programmers contribute (write, debug, analyze, support, suggest changes).
  - Arguably more secure (many eyes viewing the code).
  - Bugs found and fixed faster due to number of users/viewers.
  - Commercial companies (e.g., Red Hat) benefit from open-sourcing code (revenue from support contracts, hardware sales).

1 History

- Early computing (1950s): software generally came with source code (e.g., MIT’s Tech Model Railroad Club, Homebrew user groups, DECUS).
- 1970: Digital’s OS distributed as source code with no restrictions/copyright.
- Computer/software companies later sought to limit software use to authorized computers/paying customers.
- Releasing only binary files (not source code) helped protect code/ideas from competitors.
- By 1980: proprietary software was the usual case.

2 Free operating systems

- To counter proprietary trend, **Richard Stallman** (1984) started developing **GNU** (”GNU’s Not Unix!”): a free, UNIX-compatible OS.
- ”Free” refers to freedom of use, not price.
- Free-software movement holds users are entitled to four freedoms:
  1. To freely run the program.
  2. To study and change the source code.
  3. To give or sell copies.
  4. To give or sell copies with or without changes.
- 1985: Stallman published the **GNU Manifesto** (argues all software should be free).
- Formed the **Free Software Foundation (FSF)** to encourage free software use/development.
- FSF uses ”copyleft” (Stallman invention): licensing that gives anyone possessing a copy the four essential freedoms, with condition that redistribution must preserve these freedoms.
- **GNU General Public License (GPL):** Common license for free software; requires source code distribution with binaries, and all copies (including modified) released under same GPL.
- Creative Commons ”Attribution Sharealike” license is also a copyleft license.

3 GNU/Linux

- By 1991: GNU OS nearly complete (compilers, editors, utilities, libraries, games), but kernel not ready.
- 1991: **Linus Torvalds** (Finland) released a rudimentary UNIX-like kernel using GNU compilers/tools, invited worldwide contributions.
- Internet enabled rapid growth of ”Linux” (weekly updates, thousands of programmers).
- 1991: Linux not free (noncommercial redistribution only).

- 1992: Torvalds rereleased Linux under the GPL, making it free software (and "open source").
- **GNU/Linux:** Full OS (kernel properly called Linux, but includes GNU tools).
- Spawned hundreds of unique **distributions** (custom builds): Red Hat, SUSE, Fedora, Debian, Slackware, Ubuntu.
- Distributions vary in function, utility, installed applications, hardware support, user interface, and purpose.
- Example: Red Hat Enterprise Linux geared for large commercial use.
- Example: **PCLinuxOS** is a **Live CD** (or **LiveDVD**) – OS that can be booted/run from CD-ROM/DVD without installation on boot disk.
- Running Linux on a Windows (or other) system using virtualization:
  1. Download/install free **Virtualbox VMM** tool (`https://www.virtualbox.org/`).
  2. Install OS from scratch (CD image) or use pre-built images (`http://virtualboxes.org/images/`).
  3. Boot the virtual machine within Virtualbox.
- Alternative: free program **Qemu** (`http://wiki.qemu.org/Download/`), includes `qemu-img` for converting Virtualbox images.
- This text provides a virtual machine image of GNU/Linux running Ubuntu release (includes source code, dev tools).

4 BSD UNIX

- Longer, more complicated history than Linux.
- Started 1978 as derivative of AT&T's UNIX.
- Releases from University of California at Berkeley (UCB) came in source/binary form, but not open source (AT&T license required).
- Development slowed by AT&T lawsuit.
- 1994: Fully functional, open-source **4.4BSD-lite** released.
- Many distributions: **FreeBSD**, **NetBSD**, **OpenBSD**, **DragonflyBSD**.
- To explore FreeBSD source code: download VM image (Virtualbox), source in `/usr/src/`, kernel in `/usr/src/sys`, VM code in `/usr/src/sys/vm`.
- View source code online: `https://svnweb.freebsd.org`.
- Source code contained/controlled by a **version control system** (e.g., "subversion" `https://subversion.apache.org/source-code`).
  - Allows user to "pull" entire source code tree and "push" changes back to repository.
  - Provides history of each file and conflict resolution.
- Another version control system: **git** (`http://www.git-scm.com`), used for GNU/Linux and other programs.
- **Darwin:** Core kernel component of macOS, based on BSD UNIX, open-sourced (`http://www.opensource.apple.com/`).
  - Every macOS release has open-source components posted there.
  - Package containing kernel begins with "xnu."
  - Apple provides developer tools, documentation, support (`http://developer.apple.com`).

5 The study of operating systems

- Never been easier/more interesting to study OS.
- Open-source movement: many OS available in source and binary (Linux, BSD UNIX, Solaris, part of macOS).
- Availability of source code allows studying OS from the inside out (examining code vs. documentation/behavior).
- OS no longer commercially viable have been open-sourced (study how systems operated with fewer CPU, memory, storage resources).
- Rise of **virtualization** (mainstream, often free) makes it possible to run many OS on top of one core system.
  - Example: VMware (`http://www.vmware.com`) provides free "player" for Windows; hundreds of free "virtual appliances" can run.
  - Example: Virtualbox (`http://www.virtualbox.com`) provides free, open-source VMM on many OS.
  - Students can try hundreds of OS without dedicated hardware.
- Simulators of specific hardware also available (run OS on "native" hardware within modern computer/OS).
  - Example: DECSYSTEM-20 simulator on macOS can boot TOPS-20, load source tapes, modify/compile new TOPS-20 kernel.
- Open-source OS make it easier to move from student to OS developer (create new OS distribution).
- Access to source code limited only by interest, time, and disk space.

6 Solaris

- Commercial UNIX-based OS of Sun Microsystems.
- Originally Sun's **SunOS** based on BSD UNIX.
- 1991: Sun moved to AT&T's System V UNIX as its base.
- 2005: Sun open-sourced most Solaris code as the **OpenSolaris** project.
- 2010: Oracle's purchase of Sun left project state unclear.
- **Project Illumos:** Groups interested in OpenSolaris expanded features; basis for several products (`http://wiki.illumos.org`).

7 Open-source systems as learning tools

- Free-software movement drives legions of programmers to create thousands of open-source projects (including OS).
- Portals: `http://freshmeat.net/`, `http://distrowatch.com/`.
- Students use source code as a learning tool: modify/test programs, help find/fix bugs, explore mature systems (OS, compilers, tools, UIs).
- Availability of source code for historic projects (e.g., Multics) helps understanding and building knowledge for new projects.
- Diversity of open-source OS (e.g., GNU/Linux, BSD UNIX) with different goals, utility, licensing, purpose.
- Cross-pollination occurs (e.g., major OpenSolaris components ported to BSD UNIX), allowing rapid improvements.
- Advantages of free software and open sourcing likely to increase number/quality of open-source projects and their adoption.

8 Section glossary

Term	Definition
Free operating system	An operating system released under a license that makes its source code available and allows no-cost use, redistribution, and modification.
Open-source operating system	An operating system or other program available in source-code format rather than as compiled binary code.
Closed-source	An operating system or other program available only in compiled binary code format.
Reverse engineering	The procedure of converting a compiled binary file into a human-readable format.
GNU General Public License (GPL)	A license agreement that codifies copylefting (allowing and requiring open sourcing of the associated programs); a common license under which free software is released.
GNU/Linux (aka Linux)	An open-source operating system composed of components contributed by the GNU foundation and Linus Torvalds, as well as many others.
Distribution	A release of a version of an operating system.
LiveCD	An operating system that can be booted and run from a CD-ROM (or more generally from any media) without being installed on a system's boot disk(s).
LiveDVD	An operating system that can be booted and run from a DVD (or more generally from any media) without being installed on a system's boot disk(s).
UnixBSD	A UNIX derivative based on work done at the University of California at Berkeley (UCB).
Version control system	Software that manages software distributions by allowing contributors to "push" changes into a repository and "pull" a version of the software source-code tree to a system (e.g., for compilation).
Git	A version control system used for GNU/Linux and other programs.
Solaris	A UNIX derivative that is the main operating system of Sun Microsystems (now owned by Oracle Corporation). There is an active open source version called Illumos.
SunOS	The predecessor of Solaris by Sun Microsystems Inc.