

UNIT 10

Case Study

(2 Hrs) — Operating System (CACS251)

This unit studies three real operating systems — **DOS and Windows**, **UNIX** and **Linux** — and shows how the concepts of the earlier units (processes, memory, files, I/O, security) are put into practice.

10.1 DOS and Windows Operating System

10.1.1 MS-DOS (Disk Operating System)

MS-DOS was developed by Microsoft (based on QDOS, 86-DOS) for the IBM PC in 1981 and was the dominant PC operating system through the 1980s. It is a **single-user, single-tasking** system with a **command-line interface** and a 16-bit real-mode architecture.

Feature	MS-DOS
User / task model	Single-user, single-tasking (no multiprocessing)
Interface	Command-line (COMMAND.COM); no GUI
Memory	Real mode; 1 MB address space, 640 KB conventional memory; extended (XMS) and expanded (EMS) memory managers added later
File system	FAT12 / FAT16 with 8.3 file names; hierarchical directories from DOS 2.0
Protection	None — programs can access hardware and memory directly
Structure	Simple monolithic/layered design, no clear separation of kernel

Main components of MS-DOS

File	Function
IO.SYS (IBMBIO.COM)	Hardware-specific system initialisation and basic device drivers (BIOS extension)
MSDOS.SYS (IBMDOS.COM)	The DOS kernel — file management, memory management and system services (INT 21h)
COMMAND.COM	The command interpreter (shell); internal commands (DIR, COPY, DEL) built in
CONFIG.SYS / AUTOEXEC.BAT	Configuration files: device drivers and startup commands

Boot process: BIOS POST → loads boot sector (MBR) → loads IO.SYS and MSDOS.SYS → reads CONFIG.SYS → loads COMMAND.COM → executes AUTOEXEC.BAT → prompt C:\>.

Common commands: *DIR, CD, MD, RD, COPY, DEL, REN, TYPE, FORMAT, CHKDSK, EDIT, CLS, DATE, TIME, XCOPY, ATTRIB*. External programs are files with the extension **.COM, .EXE** or **.BAT**. Programs may stay resident using TSR (Terminate and Stay Resident) to simulate multitasking.

10.1.2 Windows Operating System

Windows began as a graphical shell on top of MS-DOS (Windows 1.0 – 3.x, 1985 – 1992) and evolved into full operating systems. There are two lines of development: the **9x line** (Windows 95, 98, ME — hybrid 16/32-bit, built on DOS) and the **NT line** (Windows NT 3.1 in 1993 → 2000 → XP → Vista → 7 → 8 → 10 → 11; Windows Server) — a true 32/64-bit, secure, portable OS designed by Dave Cutler which is the basis of all modern Windows.

Version	Year	Key features
Windows 1.0 – 3.1	1985 – 1992	GUI shell over DOS; cooperative multitasking; icons and windows
Windows 95 / 98	1995 / 1998	32-bit; preemptive multitasking; Start menu, taskbar; long file names; Plug and Play; FAT32
Windows NT 3.1 / 4.0	1993 / 1996	Fully 32-bit, portable, secure, multiuser server OS; NTFS
Windows 2000 / XP	2000 / 2001	NT technology for desktops; Active Directory; NTFS; better stability
Windows Vista / 7	2007 / 2009	Aero GUI, UAC, improved security and search
Windows 8 / 10 / 11	2012 / 2015 / 2021	Touch and tablet support, universal apps, WSL (Linux subsystem), cloud integration, 64-bit only (11)

Architecture of Windows NT-family

Windows NT uses a **hybrid (modified microkernel)** design: the microkernel and executive run in kernel mode for speed, while environment subsystems run in user mode.

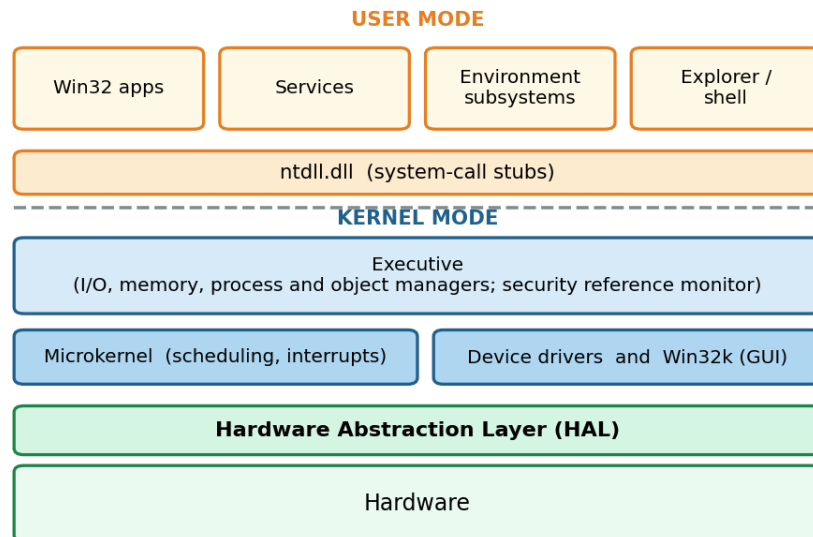


Figure 10.1 — Architecture of the Windows NT family

- **Hardware Abstraction Layer (HAL)** — isolates the kernel from hardware differences, making the OS portable.
- **Microkernel** — thread scheduling, interrupt and exception handling, synchronisation.
- **Executive** — Object Manager, Process/Thread Manager, Virtual Memory Manager, I/O Manager, Security Reference Monitor, Cache Manager, Plug-and-Play and Power managers.
- **Environment subsystems** — Win32 (main), POSIX, WSL — present applications with their expected API.
- **Registry** — a hierarchical database that stores configuration of the OS, hardware, and applications.

Key features of Windows

Area	Windows implementation
Process management	Processes and threads; preemptive, priority-based, multilevel feedback scheduling with 32 priority levels; quantum-based round robin among equal priorities
Memory management	Demand-paged virtual memory with 4 KB pages, working sets, page file (pagefile.sys), per-process private address space
File systems	FAT16/FAT32 , exFAT (removable media), NTFS (journaling, security descriptors/ACLs, compression, encryption — EFS, large files up to 16 EB), ReFS
I/O management	Layered, packet-based (IRP) I/O with device drivers; Plug and Play
Security	Logon authentication, ACLs on objects, User Account Control (UAC), BitLocker, Windows Defender, firewall

Area	Windows implementation
User interface	GUI: desktop, taskbar, Start menu, File Explorer; PowerShell and cmd for command line
Networking	TCP/IP, SMB file sharing, domains and Active Directory, Remote Desktop

MS-DOS	Windows (NT family)
Single-user, single-tasking	Multi-user (server), multitasking with preemption
Command-line	GUI plus command line (cmd, PowerShell)
16-bit real mode, 640 KB limit	32/64-bit protected mode with virtual memory
FAT file system, no security	NTFS with ACLs, encryption and journaling
Direct hardware access	HAL and drivers protect the hardware

10.2 UNIX Operating System

10.2.1 History

UNIX was created in **1969** at **AT&T Bell Laboratories** by **Ken Thompson** and **Dennis Ritchie**, as a simpler successor to the MULTICS project. In **1973** it was rewritten in the **C language** (invented by Ritchie), making it the first portable OS. Two main branches developed: **System V** (AT&T) and **BSD** (Berkeley Software Distribution, University of California). Commercial derivatives include Solaris, AIX, HP-UX, macOS (BSD-based) and FreeBSD. The **POSIX** standard (IEEE 1003) defines a common interface among UNIX-like systems.

10.2.2 Design Philosophy and Features

- **Multi-user and multitasking** (time-sharing) with virtual memory.
- **Portable** — written in C, easy to move between hardware.
- **“Everything is a file”** — devices, pipes and sockets are accessed like files (/dev/tty, /dev/sda).
- **Small tools that do one thing well** and are combined using **pipes** and redirection (ls | grep txt | wc -l).
- **Hierarchical file system** with a single root (/), mounting of file systems.
- **Shell as a user-level program** — a programmable command interpreter with scripting.
- Built-in networking (TCP/IP sockets) and strong security based on permissions.

10.2.3 Architecture of UNIX

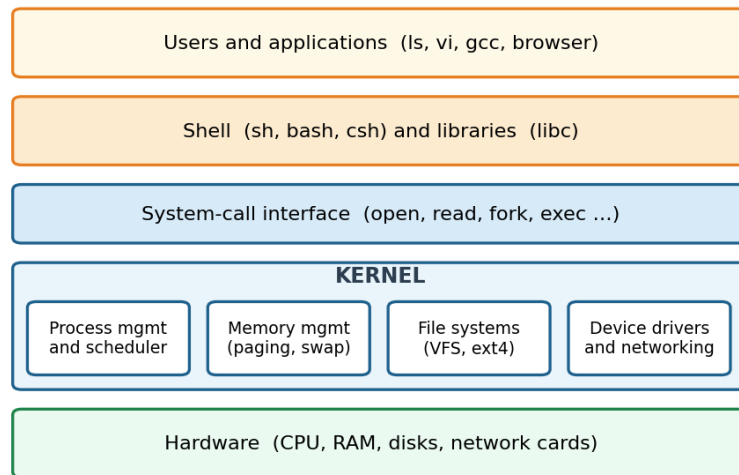


Figure 10.2 — Layered architecture of UNIX / Linux

Layer	Description
Hardware	CPU, memory, disks, terminals
Kernel	Core of the OS: scheduling, memory management, file system, device drivers, networking; runs in privileged mode. UNIX uses a monolithic kernel .
System-call interface	About 60 – 300 calls: fork, exec, exit, wait, open, read, write, close, pipe, kill ...
Shell and libraries	Command interpreter (Bourne sh, C shell, Korn ksh, bash) and the C library
Utilities and applications	Standard commands (ls, cp, grep, vi), compilers, user programs

10.2.4 Process Management in UNIX

- A process is created with **fork()** (exact copy of the parent — the child gets a return value 0, the parent gets the child's PID) and a new program is loaded with **exec()**. The parent may **wait()** for the child. The first process is **init** (PID 1).
- **Process states:** running (user/kernel mode), ready, sleeping (waiting), zombie (terminated, waiting for parent's wait()), stopped.
- **Scheduling:** priority-based **multilevel feedback queue** with round-robin time slices; priorities are recomputed dynamically (recent CPU usage lowers priority) and can be changed with **nice**.
- **IPC:** pipes, named pipes (FIFOs), signals, message queues, shared memory, semaphores, sockets.

10.2.5 Memory Management in UNIX

Early UNIX used **swapping**; modern versions (BSD, System V R4) use **demand paging** with a **page-replacement daemon** (page daemon using the **clock / two-handed clock** algorithm) and a swapper that moves entire processes out when memory is very short. Each process has a virtual address space consisting of text, data (initialised/uninitialised — bss), heap and stack segments.

10.2.6 File System in UNIX

- Files are **unstructured byte streams**; directories are files containing name → i-node number pairs.
- Every file is described by an **i-node** (index node) containing owner, group, permissions, size, time stamps, link count and pointers to data blocks — **direct** (12), **single indirect**, **double indirect** and **triple indirect** block pointers.
- Types: ordinary file, directory, character/block special (device) files, symbolic link, named pipe, socket.
- Permissions** — rwx for user (owner), group and others (e.g., rwxr-xr-x = 755); changed with **chmod**, owner with **chown**. Special bits: setuid, setgid, sticky.
- Standard directories: **/bin**, **/sbin**, **/etc**, **/dev**, **/home**, **/usr**, **/var**, **/tmp**, **/lib**, **/proc**, **/mnt**. File systems: UFS/FFS, ext2/3/4, XFS, ZFS.

Command	Purpose	Command	Purpose
ls	List directory	cp / mv / rm	Copy / move / delete
cd, pwd	Change / show directory	cat, more, less	Display files
mkdir / rmdir	Create / remove directory	grep	Search text pattern
chmod / chown	Change permission / owner	ps, top, kill	Show / control processes
who, whoami	Logged-in users	man	Manual pages
and >	Pipe / redirect output	sudo, su	Run as another user

10.2.7 Security in UNIX

Each user has a **user ID (UID)** and one or more **group IDs (GID)**; the superuser **root** (UID 0) bypasses permission checks. Authentication is by password (salted hashes in **/etc/shadow**); access control uses the rwx mode bits (and optionally ACLs); the **setuid** bit lets a program run with its owner's privileges (e.g., **passwd**).

10.3 Linux Operating System

10.3.1 History

Linux is a free and open-source, UNIX-like operating system kernel. It was started in **1991** by **Linus Torvalds**, a Finnish student at the University of Helsinki, who announced a hobby kernel for the Intel 80386. Combined with the GNU tools and utilities of Richard Stallman's **GNU Project** (compiler gcc, bash, coreutils) it forms the complete **GNU/Linux** system. The kernel is released under the **GNU General Public License (GPL)** and developed by thousands of contributors worldwide. Linux runs on everything from smartphones (Android), routers and embedded devices to servers, cloud infrastructure and all of the world's top-500 supercomputers.

Year	Event
1991	Linux 0.01 announced by Linus Torvalds
1994	Linux 1.0 — first stable release
1996	Linux 2.0 — SMP support; mascot Tux
2003	Linux 2.6 — new O(1) scheduler, preemptible kernel
2011	Linux 3.0
2015 –	Linux 4.x, 5.x, 6.x — continuous improvements (eBPF, containers, Rust support)

10.3.2 Features

- **Open source and free** — anyone may study, modify and redistribute the source.
- **Multi-user, multitasking, multiprocessor (SMP)** and **portable** (x86, ARM, RISC-V, PowerPC and more).
- **Monolithic kernel with loadable kernel modules (LKM)** — drivers and file systems can be loaded and unloaded at run time (insmod, modprobe).
- **Preemptive kernel** and the **Completely Fair Scheduler (CFS)**; real-time scheduling classes (FIFO, RR, deadline).
- **Virtual memory** with demand paging, copy-on-write, swapping, and multilevel page tables.
- **Many file systems** supported through the **Virtual File System (VFS)** layer: ext4 (default), XFS, Btrfs, FAT, NTFS, NFS.
- Strong security: permissions, ACLs, **SELinux / AppArmor**, capabilities, namespaces and cgroups (basis of containers such as Docker).
- POSIX-compliant, complete networking stack, and powerful shell (bash) scripting.

10.3.3 Linux System Components

Component	Description
Kernel	Core code that abstracts hardware, manages processes, memory, file systems and devices; runs in kernel mode.
System libraries	Standard set of functions (glibc) through which applications call the kernel — the system-call interface.
System utilities	Programs that perform specialised tasks — init/systemd, daemons (sshd, cron), login, network configuration.
Shell	Command interpreter — bash, zsh, sh, fish.
Desktop environment	Graphical interface — GNOME, KDE Plasma, Xfce, on top of X11/Wayland.
Package manager	apt/dpkg (Debian, Ubuntu), yum/dnf/rpm (Red Hat, Fedora), pacman (Arch).

A **Linux distribution** bundles the kernel, GNU tools, package manager and applications: **Ubuntu, Debian, Fedora, Red Hat Enterprise Linux (RHEL), CentOS, SUSE, Arch, Kali Linux**, and Android (modified kernel).

10.3.4 Process Management in Linux

- Process (task) creation via **fork()** and **clone()** — clone lets the child share address space, file descriptors and signal handlers with the parent, so **threads** are implemented as tasks that share resources (one-to-one model, NPTL).
- Each process is represented by a **task_struct** (the PCB) holding state, PID, priority, memory maps, open files, credentials.
- **States:** TASK_RUNNING (running/ready), TASK_INTERRUPTIBLE, TASK_UNINTERRUPTIBLE (sleeping), TASK_STOPPED, TASK_ZOMBIE.
- **Scheduling:** CFS gives each task a fair share of CPU using a red-black tree ordered by **virtual run-time (vruntime)** — the task with the smallest vruntime runs next; **nice** values (−20 to +19) weight the share.
- **IPC:** pipes, FIFOs, signals, System V/POSIX message queues, shared memory, semaphores, sockets.

10.3.5 Memory Management in Linux

- Each process has a private **virtual address space** (e.g., 3 GB user + 1 GB kernel on 32-bit x86); translation by multilevel page tables (4 levels on x86-64) with a TLB.
- **Demand paging** and **copy-on-write** for fork(); **swap space** on disk or swap file.

- Physical memory is divided into zones (DMA, Normal, High); the **buddy system** allocates page frames, and the **slab allocator** serves small kernel objects.
- Page replacement:** an approximation of LRU using active/inactive lists (a “second chance” scheme); the **kswapd** daemon reclaims pages, and the **OOM killer** terminates processes when memory is exhausted.

10.3.6 File System and I/O in Linux

- Everything is a file: devices appear under **/dev** (block and character devices) and kernel information under **/proc** and **/sys**.
- The **Virtual File System (VFS)** defines common objects — *superblock*, *i-node*, *dentry*, *file* — so that ext4, XFS, NFS and others can coexist under one tree.
- ext4:** journaling, extents, files up to 16 TB, backwards-compatible with ext2/ext3. Journaling protects against corruption on crash.
- I/O scheduling uses elevator-based algorithms — deadline, CFQ, BFQ, mq-deadline, noop (for SSD); page cache and buffer cache speed up disk access; DMA for bulk transfer.

10.3.7 Security in Linux

- User/group/other permissions, setuid/setgid, POSIX ACLs; the root account and **sudo**.
- Authentication through **PAM** (Pluggable Authentication Modules); hashed passwords in **/etc/shadow**; SSH keys.
- Mandatory access control: **SELinux**, **AppArmor**; firewall with **iptables/nftables**; namespaces and cgroups for containers; regular open-source security patching.

Feature	MS-DOS	Windows (NT)	UNIX	Linux
Users / tasks	Single / single	Multi / multi	Multi / multi	Multi / multi
Interface	CLI	GUI + CLI	CLI (+ GUI)	CLI + GUI
Kernel type	Simple	Hybrid	Monolithic	Monolithic + modules
File system	FAT	NTFS, FAT	UFS, ext	ext4, XFS, Btrfs
Source / cost	Proprietary	Proprietary	Mostly proprietary (BSD open)	Open source, free
Security	None	ACL, UAC	rxw, root	rxw, SELinux
Typical use	Old PCs	Desktop, server	Servers, workstations	Servers, cloud, Android, embedded

Key Takeaways

- ✓ **MS-DOS**: single-user, single-tasking, command-line OS (IO.SYS, MSDOS.SYS, COMMAND.COM) with the FAT file system.
- ✓ **Windows NT-family**: hybrid kernel with HAL, executive and environment subsystems; preemptive priority scheduling, demand-paged virtual memory, NTFS, ACL-based security, registry.
- ✓ **UNIX**: multi-user, multitasking, written in C; “everything is a file”, pipes, i-node-based hierarchical file system; fork/exec process model; monolithic kernel.
- ✓ **Linux**: open-source UNIX-like kernel by Linus Torvalds (1991); loadable modules, CFS scheduler, VFS, ext4, SELinux; runs on servers, cloud, Android and embedded systems.

— End of Unit 10 —

bcalink.com