OPERATING SYSTEMS · LECTURE 1

Introduction to Operating Systems

What an OS is · history · functions · architectures



Lecturer: Aidana Karibayeva

01 What is an operating system?

Software that **controls and coordinates** the hardware and runs other programs. The interface between the **user** and the **hardware**.

Convenient

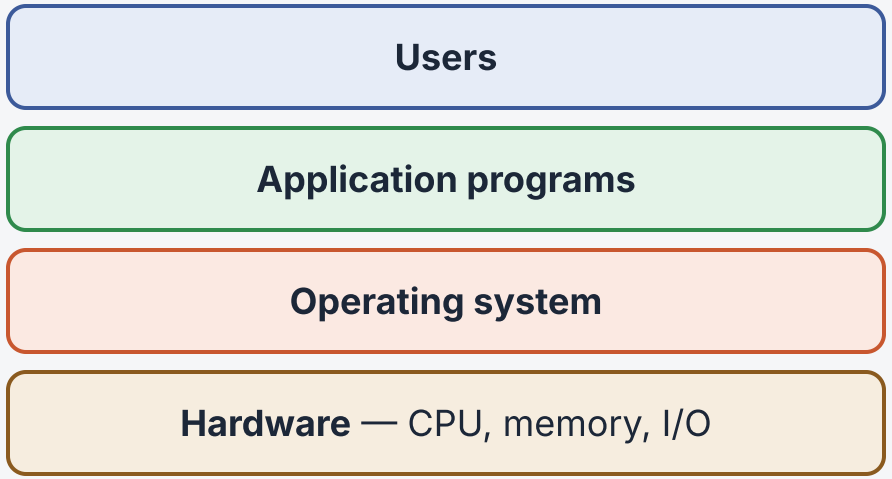
easy to use — a software engineering problem

Efficient

fair use of resources — a systems problem

`Windows` `macOS` `Linux` `UNIX` `Android` `iOS` `MS-DOS`

four components of a computer system



02

Main functions of an OS

Booting

starts the computer

Processes & CPU

create, schedule, synchronise processes

Memory

allocate, protect, swap to disk

Files

files and directories on disk

I/O devices

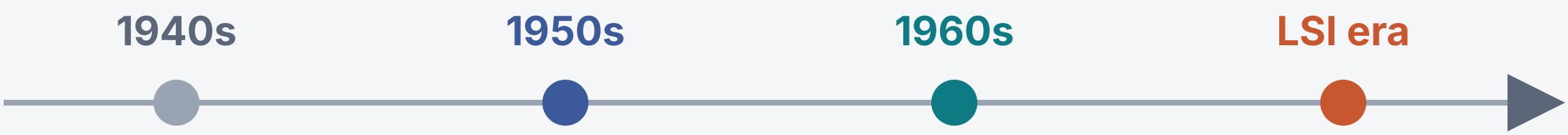
printers, disks, keyboards

Networking & security

work together, stay protected

The OS coordinates many applications and users **fairly and efficiently**.

03 History of operating systems



1st generation

no OS at all
plug boards, switches
no programming languages

2nd generation

punch cards
first OS: GM Research
for IBM 701
one job at a time

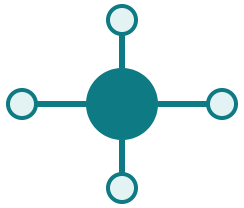
3rd generation

batch processing
several jobs at once

4th generation

microprocessors
personal computers and
workstations

04 Three views of the OS



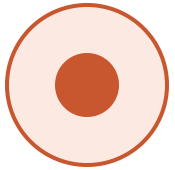
Resource allocator

allocates hardware and software resources and manages them efficiently



Control program

controls execution of user programs and I/O devices



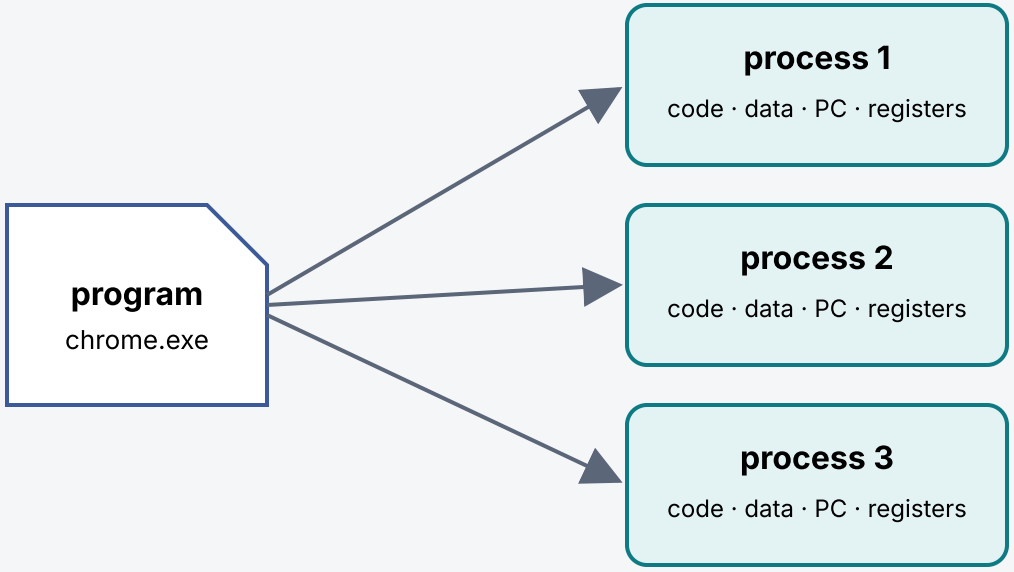
Kernel

the program that **runs all the time**; everything else is an application

Kernel data structures describe the current state — e.g. a new process gets a structure linked to all others. They live in memory, accessible only to the kernel.

05

Process ≠ program



A **process** is one instance of a program **in execution** with its full context: code, data, program counter, registers, OS resources.

> Many processes can run the same program. User programs and system programs (spoolers, file servers) are all processes.

06 What the OS must do

Processes

- create, delete, suspend, resume, schedule
- inter-process communication, synchronisation
- handle deadlock

Memory

mechanics

- track used and free memory
- protect, allocate, deallocate
- swap memory ↔ disk

policies

- when to load, how much space, when to remove

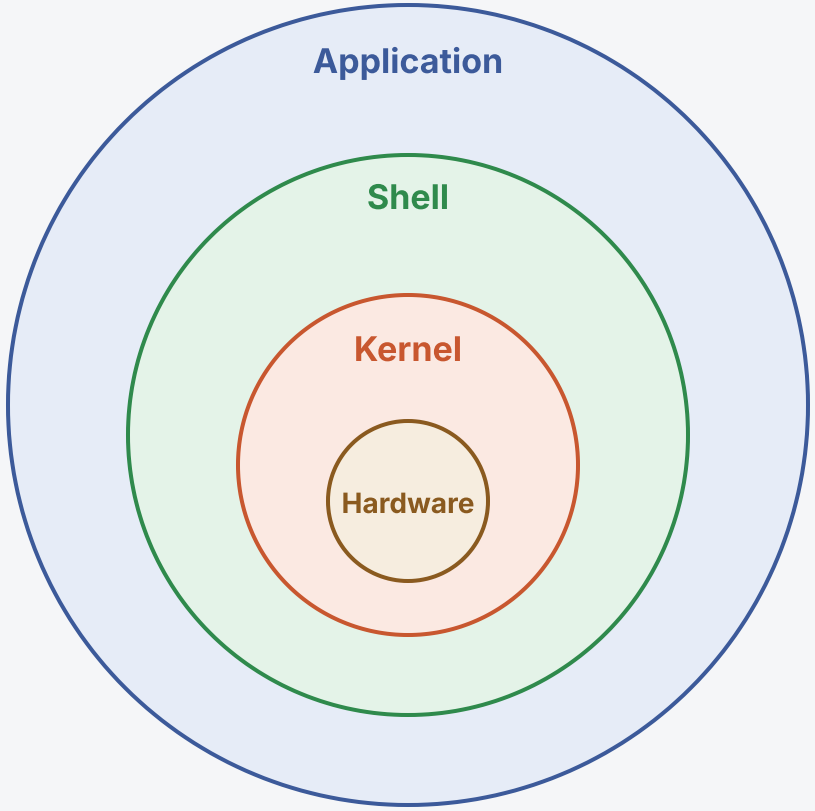
Files

- create and delete files and directories
- read, write, extend, rename, copy, protect
- backups, accounting, quotas

07 Types of operating systems

	PARALLEL	DISTRIBUTED	REAL-TIME
Memory	shared	not shared	meets **hard / soft** time constraints on processing data
Clock	shared	not shared	
Processors	many, tightly coupled	few, loosely coupled	
Single-OS look	yes	the ideal goal, rarely achieved	—

08 Parts of an OS



Application	software the user runs: Slack, Sublime Text
Shell	user interface to launch programs — command line or GUI: PowerShell, csh, ksh
Kernel	core: gives processes CPU, memory and I/O via system calls and IPC
Hardware	physical devices

? Quick check

Which part of an OS provides an interface for the user to **launch programs**?

A · Application

B · Shell

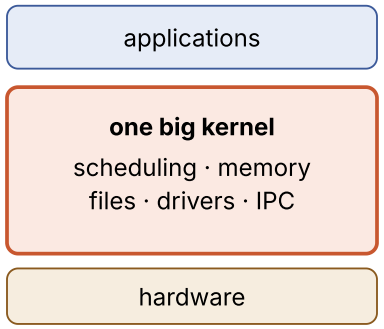
C · Kernel

D · None

09

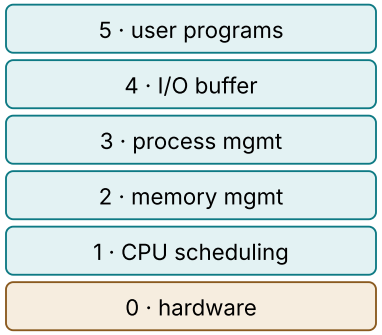
Four OS architectures

Monolithic



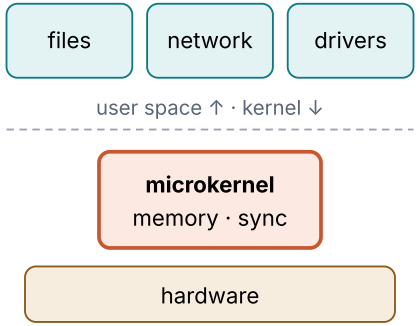
OS/360, VMX, Linux

Layered



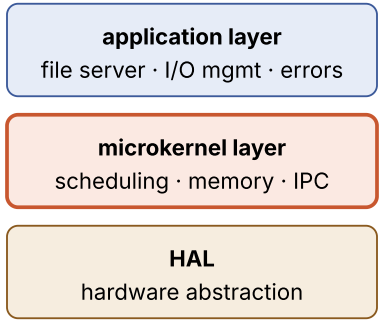
THE, Windows XP, Linux (partly)

Microkernel



modular components: Linux, Windows XP

Hybrid



Windows NT kernel

10 Monolithic architecture

All OS components live in **kernel space** and talk to each other by **function calls**.

Advantages

- scheduling, memory etc. via system calls
- everything runs in one address space
- a single static binary file

Disadvantages

- components depend on each other — one fails, **the whole system fails**
- adding a service means changing the whole OS

11 Layered architecture

5 · User programs games, browsers

4 · I/O buffer temporary storage of input

3 · Process management RR, FCFS, SJF

2 · Memory management disk ↔ RAM

1 · CPU scheduling job and ready queues

0 · Hardware keyboard, mouse, scanner

Rule: layer *i* may use layers **0 … i−1**, never layer i+1.

Advantages

modular · easy to debug and update · hardware hidden from users · each layer abstracts the others

Disadvantages

hard to order layers correctly · a request passes through every layer between → slower response

12

Microkernel and hybrid

Microkernel

Only memory management and synchronisation in the kernel; processes, network, files, devices run **outside**.

+	modular — one failure does not stop the OS
+	compact, isolated, efficient
+	new features without recompilation
−	drivers need a function call or context switch
−	services cost more than in monolithic

Hybrid

Combines the other architectures: HAL + microkernel layer + application layer.

+	each architecture contributes its services
+	easy to manage — layered approach
+	fewer layers
+	better security and protection
−	some services stay in kernel space, less critical ones move to user space

? Quick check

Which architecture runs different OS components **separately**?

A · Monolithic

B · Layered

C · Microkernel

13 Conclusion

ARCHITECTURE	KEY IDEA	EXAMPLE
Monolithic	everything in one kernel	Linux, OS/360
Layered	layers 0…n, each uses only lower ones	THE, Windows XP
Microkernel	minimal kernel, services outside	modular parts of Linux, XP
Hybrid	combination of all — most used today, better security	Windows NT

> OS components: process, memory and I/O management, error detection, control of peripheral devices.