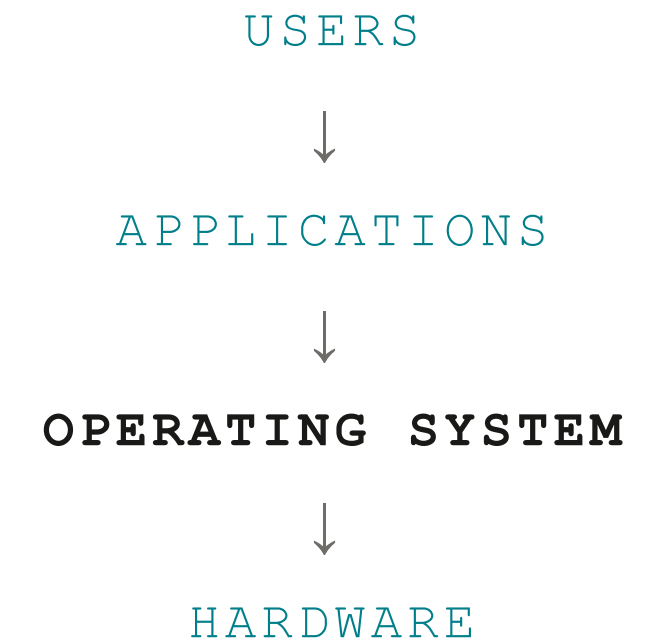


Operating Systems Fundamentals: What an OS Actually Does



Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

The Four Big Resources an OS Manages

CPU

→ Processes &
Threads

Memory

→ Memory
Management

Storage

→ File Systems

Devices

→ I/O Management

CPU → Processes and Threads

One CPU. Chrome, WhatsApp, VS Code, antivirus and hundreds of background tasks all seem to run at once. How?

WHAT

CPU scheduling — processes compete for limited CPU time.

HOW

The OS hands the CPU from process to process, very fast.

WHY

So everything looks simultaneous — on just one CPU.

MEMORIZE THIS

Processes compete; the scheduler is the referee deciding who plays next.

Memory → Memory Management

Your computer has 8 GB of RAM. Programs collectively ask for 12 GB. What happens?

Allocation

Handing out memory to each program.

Protection

Keeping one program out of another's memory.

Virtual memory

Making disk space act like extra RAM.

MEMORIZE THIS

Virtual memory: every program believes it owns all the RAM — the OS maintains the illusion.

Storage → File Systems

A disk holds billions of raw bits. Who creates the illusion of folders, filenames and directories?

011001010010...



Documents/ report.pdf
assignment.docx

The **FILE SYSTEM** turns raw bits into something humans understand.

MEMORIZE THIS

Bits don't have names — the file system is storage's illusion generator.

Devices → I/O Management

Keyboard, mouse, printer, SSD, GPU, network adapter — does Microsoft Word know how every printer in the world works?

No.

Device drivers translate one universal request into whatever that specific piece of hardware understands.

MEMORIZE THIS

Drivers are translators — one OS request, any hardware, no app rewrites.

Quick Recap: The Four Resources

CPU

Scheduling

Memory

Allocation & protection

Storage

File systems

Devices

Drivers

All four repeat one idea: share a limited resource, safely, without the user noticing the juggling.

Program $\neq$ Process

Program

Passive instructions stored on disk.

e.g. `Chrome.exe`

Process

A program currently executing.

`Chrome Process 1, 2, 3 ...`

MEMORIZE THIS

Program is the recipe; a process is the meal being cooked — cook it many times at once.

Kernel Mode vs. User Mode

User Mode

Restricted privileges. Where ordinary apps run.

Kernel Mode

Privileged access to hardware. Where the OS core runs.

Why shouldn't an ordinary Python program modify any memory location it wants?

MEMORIZE THIS

User mode asks nicely; kernel mode has the keys to the building.

System Calls: The Door Into the OS

Applications cannot normally touch hardware directly. Instead, they knock on the OS's door:



“A system call is an application asking the OS to do something privileged on its behalf.”

EXAMPLES

`open()`

`read()`

`write()`

`fork()`

`exec()`

MEMORIZE THIS

System calls are the **ONLY** legal door from an app into the kernel.

Multiprogramming vs. Multitasking

Multiprogramming

Several programs are kept loaded and ready, so the CPU always has work waiting.

Multitasking

The CPU switches rapidly between tasks, giving each a small slice of time.

MEMORIZE THIS

Programs **WAIT** their turn (multiprogramming); tasks **SWITCH** turns (multitasking).

Multiprocessing vs. Multithreading

Multiprocessing

Multiple CPUs or cores genuinely execute work at the same instant.

Multithreading

Multiple execution flows live inside one single process, sharing its memory.

MEMORIZE THIS

Cores WORK IN PARALLEL (multiprocessing); threads SHARE one process (multithreading).

Interrupts: How Hardware Gets the OS's Attention

Does the CPU keep asking the keyboard: “Have you pressed a key? Have you pressed a key?”

Usually, no.



Polling burns CPU cycles asking; an interrupt costs nothing until something actually happens.

MEMORIZE THIS

Interrupts flip the OS from “constantly asking” to “tapped on the shoulder.”

WHAT PROBLEM WERE ENGINEERS SOLVING?

A Very Short OS Evolution Story

1 No OS

2 Batch systems

3 Multiprogramming

4 Time-sharing

5 Personal computers

6 Network / distributed

7 Mobile OS

8 Cloud / virtualization

9 Modern heterogeneous

MEMORIZE THIS

History isn't trivia here — each step solved one specific engineering problem.

BEYOND WINDOWS, LINUX, macOS, ANDROID, iOS

Operating Systems Are Not Just for Laptops

Cars

Routers

Smart TVs

Robots

IoT devices

Drones

Controllers

Cloud servers

Supercomputers

MEMORIZE THIS

The OS sits between computation and the physical world — not just you and a laptop.

MOTIVATION

Why Study Operating Systems?

All code and all applications run on top of an operating system.

FND

Foundation of all
computing

EFF

Helps you write
efficient programs

SEC

Improves security and
reliability

JOB

Systems, AI, cloud,
networking,
embedded

MEMORIZE THIS

Studying the OS is studying the heart of the computer.

What Does an OS Have to Do With AI?

GPU scheduling

Which AI workload gets the GPU?

Memory management

How do large models use RAM and GPU memory efficiently?

Parallelism

Training neural networks across CPU and GPU cores.

Distributed systems

Training models across many machines at once.

MEMORIZE THIS

AI doesn't bypass the OS — it is one more demanding tenant asking for resources.

What Does an OS Have to Do With AI?

File systems

Handling massive training datasets.

Process isolation

Running AI workloads safely, side by side.

Containers

Docker and Kubernetes environments used in ML deployment.

“Even the smartest AI model depends on an OS for CPU time, memory, storage, networking and accelerator access.”

One Powerful Demonstration

CPU

→ Scheduling

Processes

→ Process management

Memory

→ Memory management

Disk

→ File systems

Network

→ I/O & networking

Users

→ Protection & security

“Almost everything you see on this screen will become a chapter in this course.”

Now Open Task Manager on Your Own Machine

Task Manager

🔍 Type a name, publisher, or PID...

— □

Processes

📄 Run new task

🛑 End task

🔄 Efficiency mode

Name	Status		16% CPU	38% Memory	0% Disk	0% Network
Apps (8)						
> 🦊 Firefox (11)	Efficiency ...	🔄	0%	636.2 MB	0 MB/s	0 Mbps
> 📄 Flow			0.1%	49.8 MB	0 MB/s	0 Mbps
> 🌐 Google Chrome (36)	Efficiency ...	🔄	8.7%	4,117.3 MB	0.1 MB/s	0.1 Mbps
> 🌐 Microsoft Edge (9)	Efficiency ...	🔄	0%	166.0 MB	0.1 MB/s	0 Mbps
> 📄 Microsoft PowerPoint			0%	97.8 MB	0 MB/s	0 Mbps
> 📄 Snagit			0.4%	168.4 MB	0 MB/s	0 Mbps
> 📄 Task Manager			2.1%	65.2 MB	0 MB/s	0 Mbps
> 📁 Windows Explorer (2)			0.8%	233.1 MB	0 MB/s	0 Mbps

Apps (8)

Eight apps you opened — far more processes underneath.

Google Chrome (36)

One program. Thirty-six live processes.

4,117.3 MB

Chrome alone is holding about 4.1 GB of RAM.

16% CPU • 38% Memory

The OS is dividing this, right now, as you watch.

MEMORIZE THIS

Everything in this one window is the OS doing its four jobs — live.

FROM ONE SCREENSHOT TO A WHOLE SYLLABUS

Every Column Here Is a Chapter of This Course

CPU 16%

→ Scheduling

Memory 38%

→ Memory management

Disk 0%

→ File systems

Network 0%

→ I/O and networking

Chrome (36)

→ Processes and threads

“End task”

→ Process control and protection

MEMORIZE THIS

Task Manager isn't a utility — it's this syllabus, rendered live on your screen.

PUTTING IT ALL TOGETHER

Your Mental Map of This Course

CPU

→ Scheduling

Processes

→ Process management

Memory

→ Memory management

Storage

→ File systems

Network

→ I/O & networking

Security

→ Protection

MEMORIZE THIS

Six panels, six chapters — that is the entire map of this course.

Memorize Pack — Vocabulary (1 of 3)

Operating System

Resource manager + abstraction layer + control system.

Resource manager

Shares CPU, memory, storage and I/O between competing programs.

Abstraction layer

Turns raw hardware into files, processes and sockets.

Program

Passive instructions sitting on disk. The recipe.

Memorize Pack — Vocabulary (2 of 3)

Process

A program currently executing. The meal being cooked.

Kernel mode

Privileged CPU mode with full hardware access.

User mode

Restricted CPU mode that ordinary applications run in.

System call

The only legal door from an application into the kernel.

Memorize Pack — Vocabulary (3 of 3)

Virtual memory

The illusion that every program owns all the RAM.

File system

The layer that turns raw bits into folders and filenames.

Device driver

Translates an OS request into hardware-specific commands.

Interrupt

A hardware signal that grabs the CPU's attention immediately.

Memorize Pack — Big Ideas (1 of 2)

OS = Manager + Translator + Guard, all in one program.

CEPC — Convenience, Efficiency, Protection, Concurrency.

Program is the recipe; a process is the meal being cooked.

Memorize Pack — Big Ideas (2 of 2)

Virtual memory: every program believes it owns all the RAM.

Bits don't have names — file systems invent folders and files.

Interrupts beat polling — wait to be tapped, don't keep asking.

That single demonstration gave you a mental map of the entire course.

Everything from here is detail — and every detail answers one of the questions we opened with.