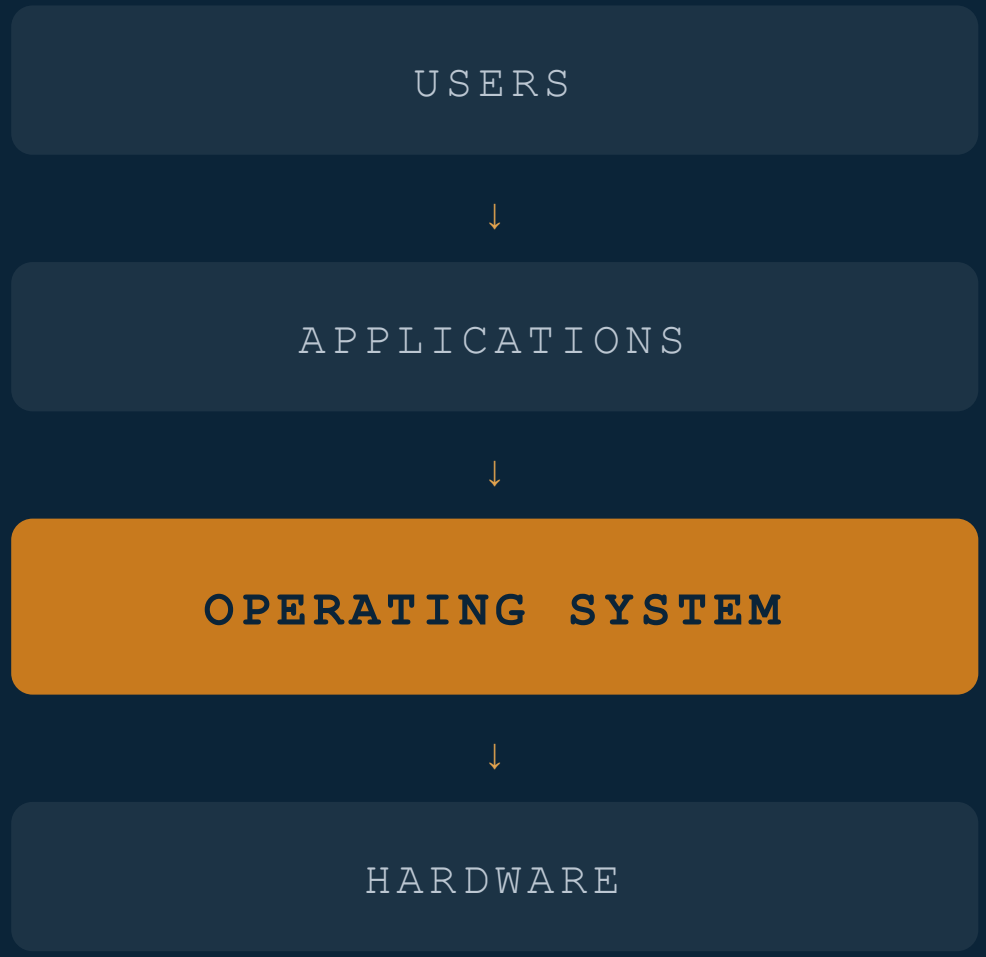
OPERATING SYSTEMS · LECTURE 01

What an Operating System Actually Does

From the power button to the process table.

Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering



Imagine Your Laptop Has No Operating System

- A CPU
- 16 GB of RAM
- An SSD
- Keyboard & screen
- Wi-Fi
- Hundreds of apps

"Here's Chrome. Run it."

What happens the instant you press Enter?

MEMORIZE THIS *No OS = raw hardware handed to a program with no rules, no sharing, no safety net.*

Who Decides…?

Where Chrome is loaded in memory?

When Chrome gets CPU time?

What happens if Spotify plays at the same time?

How Chrome reads a file from the SSD?

MEMORIZE THIS *Loading, timing, sharing, reading — four questions, one answer: the OS.*

Who Decides…?

- **How it connects to Wi-Fi?**
- **Why one crashed app doesn’t crash everything?**
- **Why one program can’t read another program’s memory?**

MEMORIZE THIS *Connecting, containing, isolating — still the OS, every time.*

Every question you just heard...

...is an Operating Systems question.

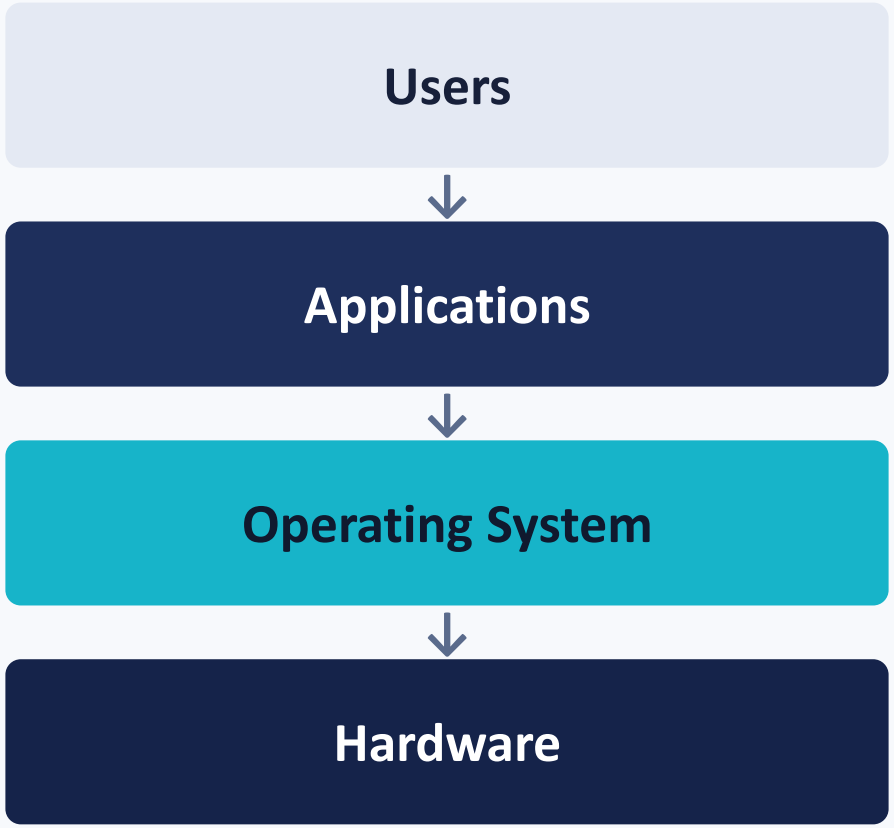
MEMORIZE THIS *Every "how does my computer just handle that?" moment is the OS at work.*

What Is an Operating System?

An OS is, at its core:

A resource manager + an abstraction layer + a control system.

MEMORIZE THIS *OS = Manager + Translator + Guard, in one program.*



What Is an Operating System?

An operating system is the program that manages a computer's hardware and provides services for application programs.

MEMORIZE THIS The OS manages the machine and serves the programs.

ما هو نظام التشغيل؟

نظام التشغيل هو البرنامج الأساسي الذي يُدير مكونات الحاسوب المادية (Hardware) وينسّق استخدامها بين البرامج التطبيقية (Application Programs) والمستخدمين. كما يعمل كبرنامج تحكّم (Control Program) لمنع الأخطاء وسوء استخدام النظام، والجزء الذي يظل يعمل دائمًا يُسمّى النواة (Kernel).

Three Roles of an Operating System

Resource Manager

CPU, memory, storage, I/O devices.

Abstraction Provider

Files, processes, virtual memory, sockets.

Control & Protection

Permissions, isolation, security.

Hardware is a city's resources. Applications are its citizens and companies. The OS is the government running the infrastructure they all share.

MEMORIZE THIS *No mayor, no rules — that's a computer with no OS.*

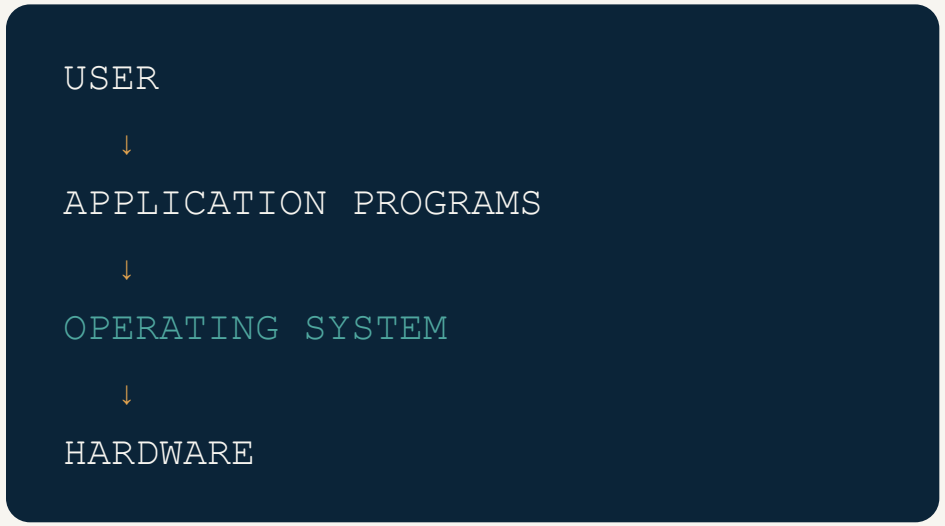
The Key Roles of an OS

- HW — Manage computer hardware
- CRD — Control and coordinate its use among programs and users
- FS — File management — create, delete, move, and control access
- CTL — Prevent errors and misuse — the control program
- KRN — Kernel — the program that is always running

MEMORIZE THIS Manage, coordinate, file, protect — and the kernel never sleeps.

Constituents of a Computer System

- USR — User — the person who interacts
- APP — Application programs — Word, browser, games
- OS — Operating system — manages and serves
- HW — Hardware — CPU, memory, I/O devices



MEMORIZE THIS Hardware + OS + applications + user = a computer system.

Program Types

Kernel

System programs

Application programs

MEMORIZE THIS Kernel runs the OS, system programs help the OS, applications help you.

The Kernel

The core of the operating system — the part that is always running.

It manages the CPU, memory and devices.

Analogy: the heart of the system. It beats whether or not you are paying attention.

MEMORIZE THIS Kernel = the always-running core.

System Programs

They support OS operations and help manage resources.

- File Manager
- Device Manager
- Control Panel
- CMD / PowerShell
- Disk Management

MEMORIZE THIS System programs = the OS's own support staff.

Application Programs

They solve the user’s problems directly.

- MS Word
- Excel
- Chrome
- Edge
- Games

MEMORIZE THIS Application programs = they serve the user.

QUICK RECAP

Three Program Types in One View

KRN	Kernel	Always running core
SYS	System programs	Support the OS
APP	Application programs	Serve the user

MEMORIZE THIS Core, support, serve — in that order.

Why Can’t Apps Control the Hardware Directly?

Convenience

Apps skip learning every device.

Efficiency

Resources get shared, not wasted.

Protection

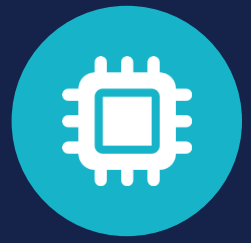
Apps can’t interfere with each other.

Concurrency

Many programs feel simultaneous.

MEMORIZE THIS *CEPC — Convenience, Efficiency, Protection, Concurrency: the OS’s four jobs.*

The Four Big Resources an OS Manages



CPU

→ *Processes & Threads*

Memory

→ *Memory Management*

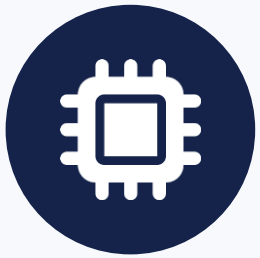
Storage

→ *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

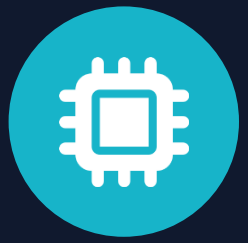
Keyboard, mouse, printer, SSD, GPU, network adapter — does Microsoft Word know how every printer in the world works?

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.

A FOUNDATIONAL DISTINCTION

Program ≠ 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

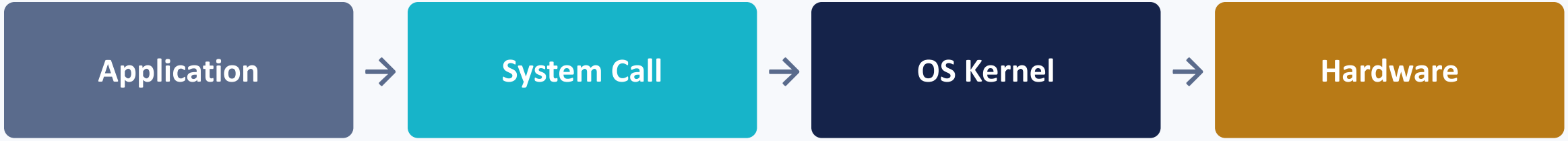
Privileged access to hardware. Where the OS core runs.

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

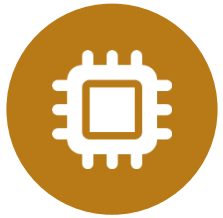
Several programs are kept loaded and ready, so the CPU always has work waiting.

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

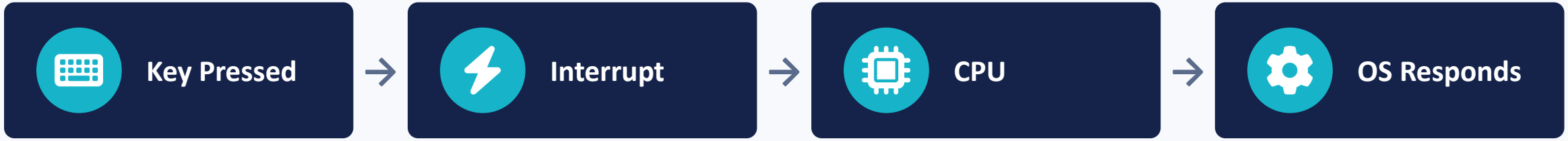
Multiple execution flows live inside one single process, sharing its memory.

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."*

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.*

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

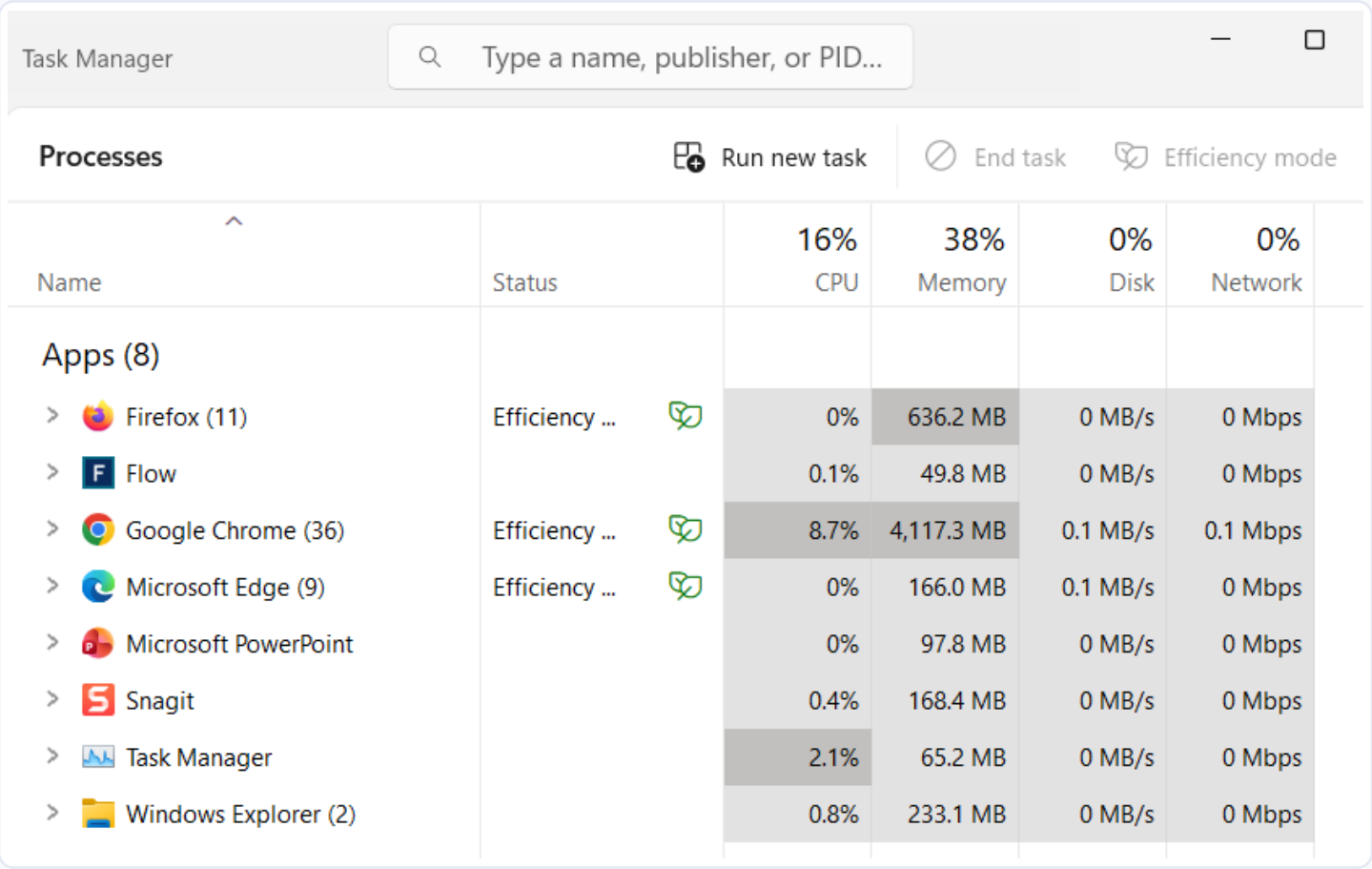
→ *I/O & networking*

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



Apps (8)

Eight apps you opened — far more processes underneath.

Google Chrome (36)

One program. Thirty-six live processes.

4,117.3 MB

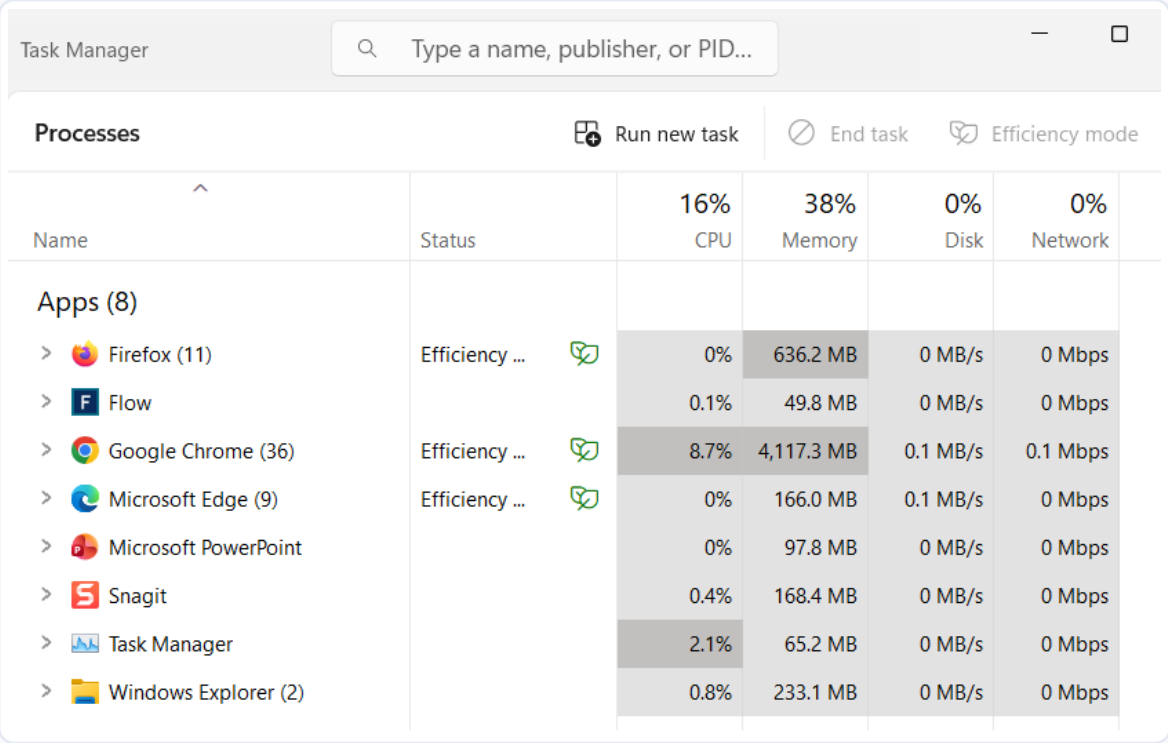
Chrome alone is holding about 4.1 GB of RAM.

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.*

Every Column Here Is a Chapter of This Course



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

CPU 16% → *Scheduling*

Memory 38% → *Memory management*

Disk 0% → *File systems*

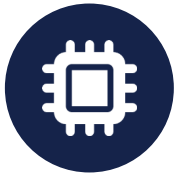
Network 0% → *I/O and networking*

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.*

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.