

Operating-System Structures

بُنَى أنظمة التشغيل

Services, interfaces, system calls, and the internal architecture of a modern kernel

PRESENTED BY

Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

One step deeper

CHAPTER 1 · INTRODUCTION

What is an OS, **why** do we need it, and **what** does it do?



CHAPTER 2 · STRUCTURES

How does it provide its services, how do programs access them, and how is it **built** internally?

IN ONE SENTENCE

Chapter 1 explains **WHAT** the OS does; Chapter 2 explains **HOW** its services are accessed, delivered, and organized internally.

01 How does the OS **provide its services** to users and applications?

كيف يقدم نظام التشغيل خدماته للمستخدمين والبرامج؟

02 How do applications **communicate with the OS** to request these services?

كيف تتواصل البرامج مع نظام التشغيل لطلب هذه الخدمات؟

03 How are its internal components **designed and organized** to deliver them efficiently, reliably, and securely?

كيف يتم تصميم وتنظيم المكونات الداخلية لنظام التشغيل لتنفيذها بكفاءة وأمان؟

Six fundamental questions

01 WHAT services does the OS provide?

Operating-System Services

02 HOW do users interact with the OS?

GUI, CLI, Shell

03 HOW do applications request OS services?

APIs and System Calls

04 HOW are programs prepared and executed?

Compilers, Linkers, Loaders

05 HOW is the OS organized internally?

Monolithic, Layered, Microkernel, Modular, Hybrid

06 HOW does the OS start, and how can we examine it?

Booting, Debugging, Monitoring

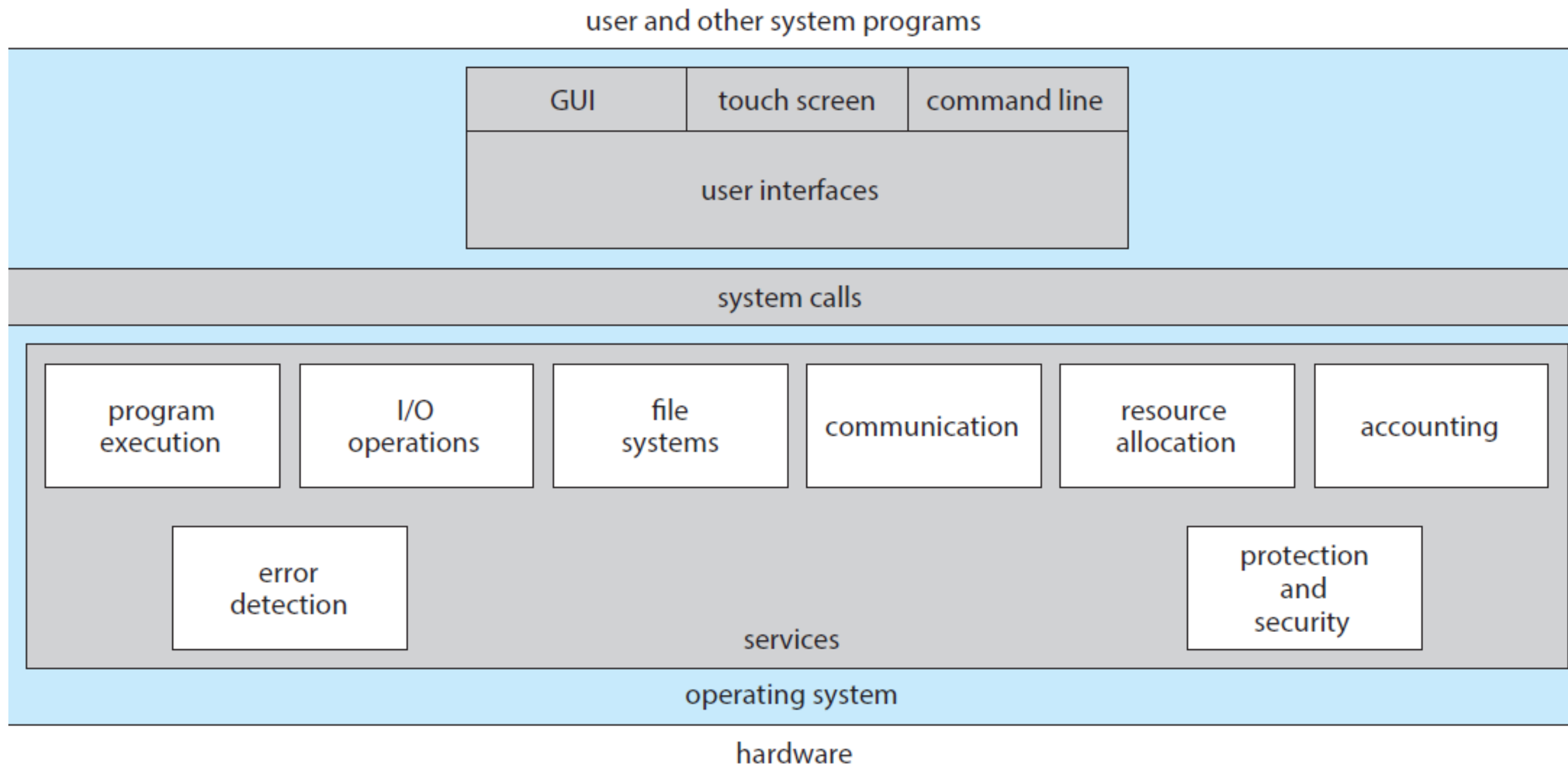
2 . 1

Operating-System Services

خدمات نظام التشغيل

The environment the operating system provides for executing programs

A view of operating-system services



Services that help the user

- **User interface** — GUI, touch screen, or command-line interface (CLI)
- **Program execution** — load a program, run it, end normally or abnormally
- **I/O operations** — the OS mediates device access for efficiency and protection
- **File-system manipulation** — create, delete, read, write, search, manage permissions
- **Communications** — shared memory or message passing, same host or across a network
- **Error detection** — constant detection and correction in CPU, memory, I/O, and user programs

Services that keep the system efficient

- **Resource allocation** — CPU cycles, main memory, file storage, and I/O devices are distributed among concurrent processes; CPU-scheduling routines weigh core count, process needs, and CPU speed
- **Logging** — record which programs use which resources, for accounting or for usage statistics that guide reconfiguration
- **Protection and security** — protection controls all access to system resources; security authenticates users and defends external I/O devices from invalid access

“A chain is only as strong as its weakest link.”

Six fundamental questions

01 WHAT services does the OS provide?

Operating-System Services

02 HOW do users interact with the OS?

GUI, CLI, Shell

03 HOW do applications request OS services?

APIs and System Calls

04 HOW are programs prepared and executed?

Compilers, Linkers, Loaders

05 HOW is the OS organized internally?

Monolithic, Layered, Microkernel, Modular, Hybrid

06 HOW does the OS start, and how can we examine it?

Booting, Debugging, Monitoring

Command Interpreters and Shells

Shell مُفسِّر الأوامر والـ

How a typed command becomes a running program

THE CENTRAL QUESTION

Does the Shell execute every command *itself* —
or does it sometimes launch *another program*?

هل كل الأوامر التي بنكتبها في الـ **Terminal** بينفذها الـ **Shell** بنفسه، ولا فيه أوامر يحتاج يشغل لها برامج خارجية؟

A program that interprets your commands

A **Shell** receives commands, **interprets** them, and either **executes** them directly or **launches** another program.

```
$ ls
```

الـ **Shell** مش شرط ينفذ كل حاجة بنفسه — أحياناً دوره إنه يستقبل الأمر ويشغل البرنامج المناسب.

User

Types `ls`

Shell — command interpreter

Interprets it, locates the program

External program `ls`

Asks the OS for directory information

Result

Files and directories displayed

Ordering at a restaurant



ANALOGY Some orders the employee answers alone; others go to the **kitchen**.

TERMINAL	SHELL
Window for typing commands	Interprets and executes them
Displays text and results	Parses syntax, launches programs
e.g. GNOME Terminal	e.g. Bash

الـ **Terminal** هو الشباك، والـ **Shell** هو الموظف اللي بي فهم طلبك.

Many shells — one kernel

`cs``h`

C Shell

Scripting syntax
influenced by C

`ba``sh`

Bourne-Again Shell

Most widely used in
Linux

`ks``h`

Korn Shell

Interactive features,
advanced scripting

`zs``h`

Z Shell

Widely used on macOS

Changing the Shell ≠ changing the OS

Different shells can run on the same Linux system — **the kernel stays the same.**

يمكن استخدام أكثر من **Shell** على نفس نظام التشغيل، من غير ما تغيّر الـ **Kernel**.

Commands that live inside the Shell

Implemented **inside the Shell** — executed with **no external program**

```
$ cd Documents
```

Why must `cd` be built in?

It changes the Shell's **own** directory — a separate process could only change its own.

COMMAND	FUNCTION
<code>cd</code>	Change directory
<code>pwd</code>	Show current directory
<code>echo</code>	Display text
<code>export</code>	Set environment variables
<code>history</code>	Show previous commands

ANALOGY Asking the receptionist for your **table number** — answered on the spot.

الـ **Built-in** : أمر ينفذه الـ **Shell** بنفسه.

Separate programs the Shell launches

```
$ rm file.txt
```

- 1 Shell recognizes `rm` as an **external** program
- 2 Finds the executable — often `/usr/bin/rm`
- 3 Launches `rm`, passing `file.txt` as an argument
- 4 `rm` requests deletion via a **system call**
- 5 Kernel checks **permissions** — performs it or returns an error

User

`rm file.txt`

Shell

Locates and launches `rm`

External program

Requests removal via a system call

Kernel

Checks permissions, performs it

الـ **Shell** مش هو اللي بيحذف الملف — بيشغل برنامج **rm**، والبرنامج
يطلب من الـ **Kernel**.

Built-in vs external commands

BUILT-IN COMMANDS	EXTERNAL COMMANDS
Implemented inside the Shell	Stored as separate executables
Executed directly by the Shell	Launched by the Shell
No external program needed	Another program must start
Example: <code>cd</code>	Examples: <code>rm</code> , <code>ls</code> , <code>cat</code>

Built-in : الموظف ينفذ الطلب بنفسه

External : الطلب يروح للمطبخ

“New commands are just new files”

- You can add new commands without modifying the Shell or Kernel.

FILE: `welcome`

```
#!/bin/bash
echo "Welcome to Operating Systems!"
```

```
$ chmod +x welcome
$ ./welcome
Welcome to Operating Systems!
```

Place it in a directory on the `PATH` — then just type `welcome`

- No change to the **Shell**
- No change to the **Kernel**
- This is **extensibility**

نقدر نضيف أوامر جديدة بسهولة من غير ما نغيّر نظام التشغيل نفسه

—وده معنى **Extensibility**

Connecting programs with a pipe

</> **Bash**

```
ps aux | sort -nrk 3,3 | head -n 5
```

The **|** **pipe** sends one program's output into the next program's input

ps aux

List process information

sort -nrk 3,3

Sort by CPU usage, descending

head -n 5

Show the top five

الـ **Pipe** بيخلى مخرجات برنامج تبقى مدخلات لبرنامج تاني.

Four concepts — and one lesson

Terminal

Where you type and see output

Shell

Interprets and arranges execution

Built-in

Run by the Shell itself

External

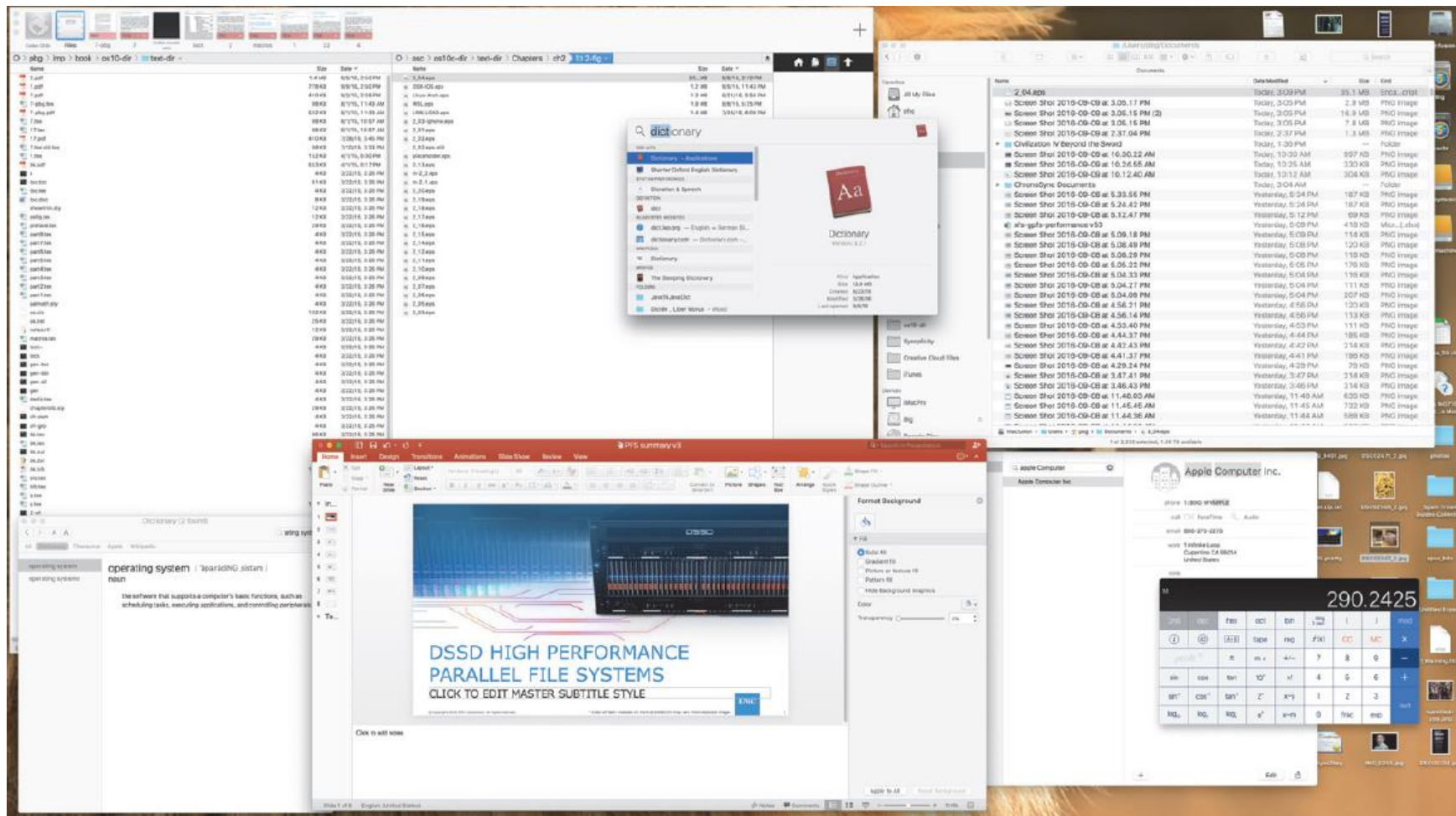
A separate program the Shell launches

The Shell is **not** the Kernel.

It is a **user-level program** between users and operating-system services.

الـ **Shell** مش هو الـ **Kernel** — هو برنامج بيسهّل تفاعل المستخدم مع خدمات نظام التشغيل.

The desktop metaphor



- Mouse-based window-and-menu system: pointer, icons, files, and folders on a desktop
- First GUI on the Xerox Alto, 1973; widespread with the Apple Macintosh in the 1980s
- Open-source desktops KDE and GNOME bring GUIs to Linux and UNIX systems

Gestures instead of mouse and keyboard



- A CLI or a mouse-and-keyboard system is impractical for most mobile devices
- Users press and swipe fingers across the screen; the keyboard itself is simulated on screen
- Both iPad and iPhone use the **Springboard** touch-screen interface

Mostly a matter of preference

Command line

Preferred by system administrators and power users: faster access, some functions available only there, and repetitive tasks become **shell scripts** interpreted by the shell

GUI and touch

Most Windows users never open a shell; macOS offers both Aqua and a command line; on iOS and Android, nearly all interaction is through the touch screen

The user interface is substantially removed from the actual system structure — designing it is not a direct function of the operating system. From the OS point of view, user programs and system programs are not distinguished.

2 . 3

System Calls

استدعاءات النظام

The interface to the services made available by an operating system