

Resource Management

Section 1.5 — The operating system is a resource manager

PROCESS

MEMORY

FILES

STORAGE

CACHE

I/O

Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

The OS Is the Hotel Manager

THE MANAGER DECIDES

- Who gets a room?
- Who uses the elevator?
- Who accesses a file?
- Who is served first?
- Who stays out of others' rooms?

THE ANALOGY

Processes	guests
Memory	rooms
Files	guest records
Storage	warehouse
Cache	reception shelf
I/O	elevators, phones

SAME CYCLE FOR EVERY RESOURCE



Six Resources the Operating System Manages

1.5.1

Process management

Create, schedule, suspend, synchronize, and connect processes.

1.5.2

Memory management

Track, allocate, and move programs and data in RAM.

1.5.3

File-system management

The file abstraction, directories, and access control.

1.5.4

Mass-storage management

Secondary storage for daily use, tertiary for backup.

1.5.5

Cache management

Faster copies — and the consistency problem they create.

1.5.6

I/O system management

Hiding hardware details behind buffers, interfaces, and drivers.

A program is **not** a process.

Program — passive

A **file on disk**: just instructions. The **recipe** ,
written on paper.

Process — active

An **instance of a program in execution** ,
holding resources. The **cooking** itself:
ingredients, tools, and effort.

Examples: a compiler running, a word processor in use, a social media app on a phone.

Resources Needed by a Process

CPU time

Memory
physical and logical

Files

I/O devices

Initialization
input

a browser process
gets a URL to load

A process may need to access **files** – **Browser** Example
configuration files – cookies - cache files - downloaded files - history
databases.

initial data or
parameters
supplied when the
process begins.

Resources are allocated to the process while it is running . When the process finishes, the operating system reclaims (يستردّ الموارد) them for reuse.

What Is a Thread (خيط)?

مسار تنفيذ داخل العملية. The **smallest sequence of execution** inside a process.

EACH THREAD OWNS

- Program counter
- Registers
- Stack

THREADS SHARE

- Code & data
- Heap
- Open resources

browser process

Thread 1 → User interface
Thread 2 → Network communication
Thread 3 → Rendering
Thread 4 → Background work

Process = workplace. **Thread** = worker.

ANALOGY

A Hospital Department

The department has shared resources:

- Patient records
- Rooms
- Medical equipment
- Database
- Supplies

DOCTORS AT WORK

Doctor 1 → examines A

Doctor 2 → reviews B

Doctor 3 → treats C

Different work, **same shared resources.**

THE MAPPING

Department	Process
Doctor	Thread
Current task	Program counter
Records, equipment	Shared resources

Each **doctor** is a **thread**.

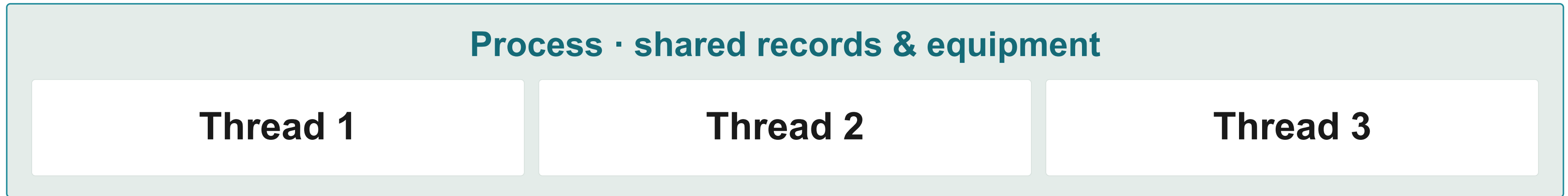
Single-Threaded vs. Multithreaded

SINGLE-THREADED · ONE DOCTOR



One process + one thread = **one execution path**.

MULTITHREADED · SEVERAL DOCTORS



Multiple execution paths inside the same process.

Why Its Own Program Counter?

Every thread may be at a **different step**.

Thread 1

`instr 100`

Thread 2

`instr 450`

Thread 3

`instr 820`

“Where am I in my work?”

A thread is an **independent execution path** inside a process, **sharing its resources**.

Process Characteristics

Single-threaded process

One program counter; instructions execute **sequentially, one at a time**.

Multithreaded process

Multiple program counters, one per thread — several execution flows inside the same process.

A system is a **collection of processes**: **OS processes** run system code, such as daemons; **user processes** run user programs.

Concurrently

T1 → T2 → T3 → T1 →

On **one CPU**, via **time-sharing**.

(تشغيل متزامن على معالج واحد) عن طريق تقاسم الوقت

In parallel

Core 1 → Thread 1
Core 2 → Thread 2
Core 3 → Thread 3

Across **multiple CPUs or cores**, genuinely **at the same time**.

OS Responsibilities: Processes - OS manages the entire life cycle of processes

01 **Create and delete** processes, both system and user

02 **Schedule** processes and threads on the CPUs

03 **Suspend and resume** processes

04 Provide **synchronization** — coordination between processes

05 Provide **communication** between processes

Double-click Chrome

→ OS creates a Chrome process
→ allocates memory and resources
→ places it in the ready queue.

When the process finishes:

- OS terminates it and reclaims its resources.

- Synchronization = controlling the order of access to shared resources.

Communication = allowing processes to exchange data and signals safely – (shared memory, pipes, message passing, sockets.)

Main Memory Is Central

Main memory is a **large array of bytes**, each with its own address, **shared by the CPU and the I/O devices**.

Instruction-fetch cycle

The CPU **reads the next instruction** from memory.

Data-fetch cycle

The CPU **reads and writes data** in memory.

Direct access only here

To process data from disk, that data must **first be loaded into RAM** — and **instructions must be in memory to execute** .

A program must be **mapped to absolute addresses and loaded into main memory** . During execution it reaches instructions and data through those addresses; when it finishes, **its space is freed for the next program** .

When it terminates:

Process ends → OS releases its allocated memory → memory becomes available to other processes

Why Memory Management Is Needed

Modern systems keep **multiple programs in memory at the same time** — multiprogramming. Managing that memory is what makes it work.

Multiprogramming

Improve CPU utilization

Another process is **always resident and ready** , so the CPU **need not idle**.

Multitasking – Time sharing

Improve response time

Interactive work gets attention quickly , because switching does not wait on disk.

Many different schemes exist; the effectiveness of any one depends on the **hardware design of the system** and on **what the workload needs**.

A server, smartphone, gaming PC, and embedded system may have very different requirements.

OS Responsibilities: Memory

01

Tracking usage

Which parts of memory are **in use**, and **which process is using them** .

02

Allocating and freeing

Memory space **granted as processes start** and **released as they finish**.

03

Deciding movement

Which processes or data **move into and out of memory — swapping and paging** .

What Is Abstraction?

Abstraction = hide the hard details, show a **simple view**.

التجريد

File abstraction: the OS hides physical storage and lets us use **simple named files** .

PHYSICAL STORAGE

where data **really lives**

FILE

the **simple logical object** we use

Physical storage → File: **the OS bridges the gap.**

THE SIMPLIFICATION

Sectors Become a File

INSTEAD OF THINKING

“My data is in **sector 1502** on this SSD,
and part is in **block 8900...**”

HDD

SSD

USB

CD

WE SIMPLY THINK

“I have a file called `report.docx`.”

Wherever it lives, **the user doesn't need to care.**

What Is a File?

A named collection of related information.

مجموعة بيانات مترابطة لها اسم

`photo.jpg`

image

`program.c`

source code

`music.mp3`

audio

`app.exe`

compiled code

Also: **text documents, video, databases.**

What You See vs. What Happens

Files, names, folders — **not** sectors, tracks, blocks.

الرؤية المنطقية

YOU SEE



YOU DON'T SEE



The OS hides all of that complexity.

Simple Operations on Files

Create

إنشاء

Open

فتح

Read

قراءة

Write

كتابة

Close

إغلاق

Delete

حذف

```
open("lecture.pdf")
```

**OS finds location****Accesses device**

The program **never talks directly** to the SSD hardware.

ANALOGY

The Library

YOU ASK

“Give me **Operating System Concepts** .”

YOU DON'T ASK

“Shelf 7, row 3, position 11...”

Book title → **File name**
Library system → **File system**
Shelf location → **Physical storage**

The **library system** works like the **file system**.

Remember This

File	what the user sees
-------------	--------------------

Storage blocks	where data actually lives
-----------------------	---------------------------

File system	links the two
--------------------	---------------

The OS **hides physical storage** and lets us work with **simple named files**.

نظام التشغيل يخفي تفاصيل التخزين، ونتعامل مع ملفات لها أسماء

Storage Devices and File Types

Storage devices

Secondary — HDDs and SSDs, the main long-term storage.

Tertiary (تخزين أبطأ، غالبًا للنسخ الاحتياطي والأرشفة) — tapes and optical disks, slower, often used for backups.

Each device has its own **access speed, capacity, data-transfer rate** , and access method: **sequential or random** .

File types

Programs — source code or compiled object code.

ملف يحتوي تعليمات أو كود البرنامج

Data — text, numbers, images, audio, video, binary.

ملف يحتوي بيانات يستخدمها البرنامج أو المستخدم

Structure may be **free-form**, like a text file, or **rigidly formatted** , like an MP3 or a database record.

Directories and Access Control

Directories

Organize files **hierarchically** so they are easier to use, and support **navigation and management** .

Directory = logical container used to organize files.

Documents
└─ Operating Systems
 ├─ Lecture1.pdf
 ├─ Lecture2.pdf
 └─ Assignments

Directories make files easier to:

find — البحث

group — التجميع

navigate — التنقل

manage — الإدارة

Access control

Needed in **multi-user systems**: it specifies **permissions** — read, write, append, **execute** — for each user.

Student A: read only

Instructor: read + write

System administrator: full access

When multiple users share access to files, the OS decides **who may access which file, and in what way** .

OS Responsibilities: Files

The OS lets users work with simple files and folders, while it handles the difficult physical storage details behind the scenes.

01 **Create and delete files**

02 **Create and delete directories**

03 **Provide primitives** (العمليات الأساسية) — the basic operations: **open, close, read, write**

04 **Map files onto physical mass storage** Logical File → File System → Physical Disk/SSD Blocks

05 **Back up files** (النسخ الاحتياطي) on stable, non-volatile media

Mass-Storage Management

Main memory is **volatile**, so it needs backup. Programs and data are stored long-term on **secondary storage**: HDDs (Hard Disk Drive), SSDs (Solid-State Drive), and NVM devices.

Almost **every program** — compiler, browser, game — depends on these devices as **both the source and the destination** of its processing.

The **speed and efficiency of storage management directly affect overall system performance**: the **entire speed of operation of a computer may hinge on this subsystem** and its algorithms.

■ كفاءة وسرعة إدارة التخزين تؤثر مباشرة على أداء النظام بالكامل.

Slow storage can make the entire computer feel slow.

Two Types of Storage

Secondary — essential

HDDs, SSDs, and **NVM (non-volatile memory)**.

Used **daily** for programs and data; the OS **must manage it efficiently** .

Tertiary — optional

Magnetic tapes, CD, DVD, Blu-ray: **slower, cheaper, often higher in capacity** .

Used for **backups, archiving, and rarely used data** — managed either by the OS or by applications.

Tertiary storage is **not crucial to system performance**, but it still **must be managed**.

OS Responsibilities: Secondary Storage

Mounting and unmounting الربط والفصل

Making a device **accessible**, or **removing it safely**.

Free-space management

Tracking **which space is unused** .

- Which disk blocks are free?
- Which blocks are already used?

Storage allocation

Deciding **where files and programs are placed** .

- Insert USB → OS recognizes it → mounts it → files become accessible.
- Why safely? Because the OS may still have unwritten data in memory.

Disk scheduling

Ordering requests to **minimize seek and rotation time** .

Partitioning

Dividing a disk into **logical sections**.

Protection

Ensuring **security and controlled access** .

OS Responsibilities: Tertiary Storage

01

Mounting and unmounting
media in devices

02

Allocating and freeing
devices for **exclusive use**
by processes

03

Migrating data between
secondary and tertiary
storage

Some operating systems take on this task; **others leave tertiary-storage management to application programs .**

Caching is a **speed booster**.

مُسْرِع الأداء

01

Data usually lives in a **slower system** — disk, or main memory.

02

When needed, a **copy is placed in a faster system** : the **cache**.

03

Next time: **found** → **use it from the cache** ; **not found** → **fetch from the source** and cache it for later.

Types of Caches

Instruction cache

Stores the **instructions expected next** , so the CPU does not wait **several cycles** for a memory fetch.

- Instruction Cache (I-cache) keeps copies of instructions that the CPU is likely to need soon.

▪ يحتفظ بالتعليمات المتوقعة تنفيذا قريباً

Data cache

Stores **frequently used data** in the memory hierarchy.

- Suppose a program repeatedly accesses:
array[0], array[1], array[2], ...
- Rather than going to RAM every time, recently/frequently needed data can remain in cache.

Note: instruction and data caches are usually **hardware-only** — **outside the control of the operating system** .

Characteristics of Various Types of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Cache Management Challenges

Caches are **small**, so something must be **evicted to make room** — which makes the **replacement policy** a central design problem.

FIFO

Evict the entry that **arrived first**.

LRU

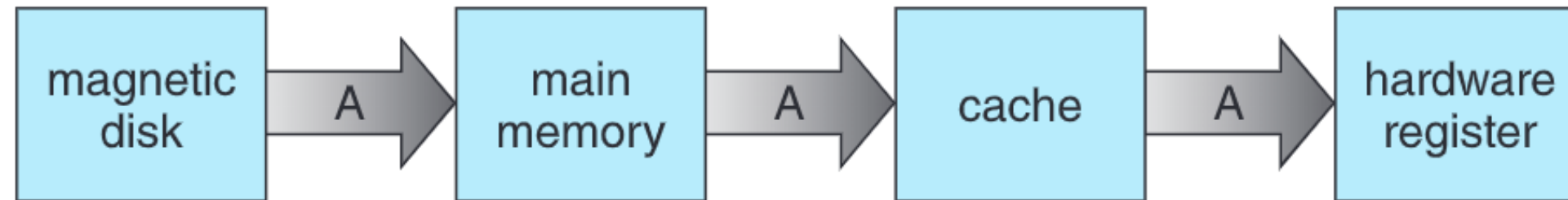
Evict the **least recently used** entry.

LFU

Evict the **least frequently used** entry.

Good policy plus the correct cache size gives a big performance gain.

Hierarchical Storage Example



When the program needs A, the value moves closer to the CPU:
 Disk → Main Memory (RAM) → Cache → Register

- Disk / SSD — Block
 - Main Memory (RAM) — Page / Frame
 - Cache — Cache Line / Cache Block
 - CPU Register — Word / Register Value
-
- Many copies may exist, but the system must preserve consistency (الاتساق) among them.

University document

Imagine an official student grade.

- The master record is in the university archive: **Disk**
- A copy is brought to the department office: **RAM**
- A copy is placed on the professor's desk: **Cache**
- The professor writes the current grade on a small note in his hand: **Register**
- If the professor changes the grade from 80 to 85, initially only the note may show 85.
- Later the desk copy, department record, and official archive **must be updated**.

Issues with Multiple Copies

Single process

No problem: the access always uses the **top-most copy** (closest to the CPU) in the hierarchy.

Multitasking

One CPU, many processes: the system must ensure each process sees the **latest updated value** .

Multiprocessor

Each CPU keeps a **local cache**, so **cache coherency** is required — usually by hardware protocols such as **MESI**.

Distributed

File replicas on different machines must be synchronized — **replica consistency** , handled by the distributed system.

■ ضمان أن جميع الـ Cache في الأنوية أو المعالجات ترى نفس القيمة الصحيحة والمحدثة للبيانات المشتركة.

■ **Cache coherency** and **replica consistency** are the **two major challenges** the hierarchy creates.

- **Cache coherency** = same data across CPU caches
- **Replica consistency** = same data across different computers

The I/O Subsystem

One of the OS's main jobs is to **hide hardware details** : users and applications should not need to know how a printer, a disk, or a keyboard works internally.

Memory-management component

Buffering — temporary storage while data moves between device and memory.

Caching — copies of frequently used data for faster access.

Spooling — queueing jobs, such as printing, so a device handles them **one at a time**.

General device-driver interface

A **uniform way** for the OS to interact with devices, **hiding the differences** between them.

Device drivers

Software modules for **specific hardware**. **Only the driver** knows the device's commands, signals, and behaviour.

Interrupt handlers alert the CPU when I/O is done, **so it does not wait**.

The I/O Subsystem

How the OS manages devices while hiding their hardware details

What, Not How

Applications say **what they want**, not **how the hardware** must do it.

THE PROGRAM SAYS

“Print this document.”

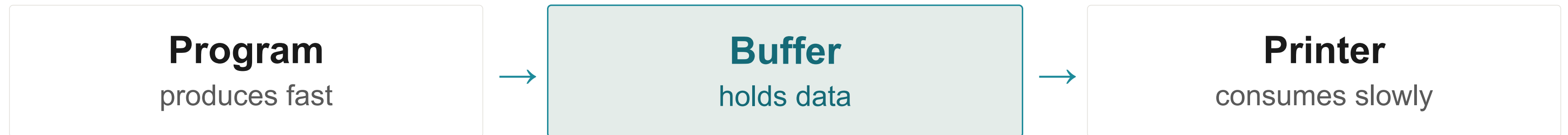
IT DOESN'T NEED TO KNOW

electrical signals, registers, timing,
internal commands

The **OS and the device driver** handle those details.

A Temporary Holding Area

Buffer = temporary memory between components of **different speeds** .
التخزين المؤقت



TYPICAL EXAMPLES

Keyboard input

Network receive

Disk I/O

The program **doesn't wait** for every character to print.

Keep a Copy Close By

Cache = a copy of data that may be **needed again soon**.

التخزين المخبئي

WITHOUT CACHE

Disk → RAM **every time**

Buffering helps data **move**.

WITH CACHE

1st access → **disk**
Later → **faster cached copy**

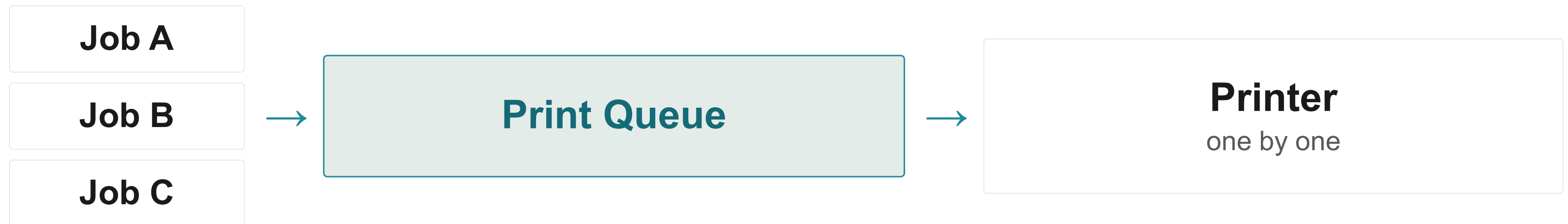
Caching helps data be **reused faster**.

One Device, Many Jobs

Spooling = complete jobs placed in a **queue** for later device processing.

تجميع المهام في طابور وانتظار تنفيذها بالتتابع

Spooling is especially useful when many programs want to use a device that can handle only one job at a time.



Buffer = temporary data during transfer

Spool = queue of jobs waiting for a device

One Interface for All Devices

HP printer

Canon printer

SSD

Keyboard

Network card

Uniform: `open()` · `read()` · `write()` · `close()`

واجهة موحدة

Application



OS I/O Interface



Device Driver



Device Controller



Device

Hides differences between devices — another form of **abstraction** (التجريد).

The Translator

Software for a **specific device** or device family.

برنامج التعريف / برنامج تشغيل الجهاز

THE DRIVER KNOWS

device commands registers · status signals
error codes communication protocol

ANALOGY

OS speaks English

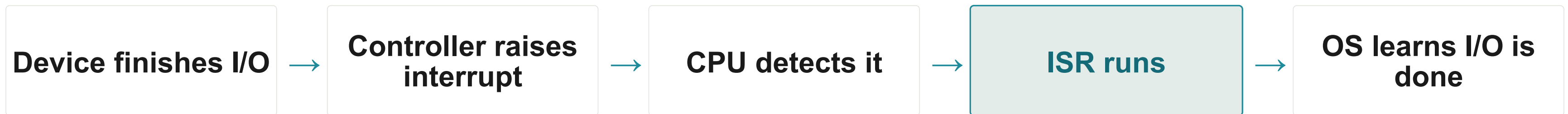
Printer speaks Japanese

Driver translates

OS speaks a **general language** → driver translates to the **device's language**.

The Device Says “Done”

Devices signal I/O completion through **interrupts**; the OS handler processes the event — **no polling needed**.



POLLING

CPU keeps asking “**Finished?**”

INTERRUPT

Device tells CPU “**I'm finished.**”

The **device/controller** raises the interrupt — not the handler.

The Airport

Buffering

Passengers wait in a **boarding area**

Caching

Frequent documents kept **close to the desk**

Spooling

Departures placed in a **queue**

Device interface

Standard procedures for all airlines

Device driver

Each **airline rep** knows its own rules

Interrupt

Gate notifies: “**Boarding complete.**”

Remember This

Device drivers Uniform interface

**Buffering · Caching Spooling ·
Interrupts**

The OS **hides device complexity** using drivers and a uniform interface, while buffering, caching, spooling, and interrupts **make I/O efficient**.

Key Takeaways

- 01 The operating system is a **resource manager**: CPU time, memory, files, storage, and I/O devices.
- 02 A **program is passive, a process is active** — the recipe versus the cooking.
- 03 Only **main memory is directly accessible** to the CPU, so multiprogramming makes **memory management unavoidable** .
- 04 **Caching enhances performance but duplicates data** — hence replacement policies, **cache coherency** , and **replica consistency**.
- 05 The **I/O subsystem hides device peculiarities** behind buffers, a uniform interface, and drivers. **Next:** Section 1.6, Security and Protection.