



Process Scheduling

Lecture 02 · Section 3.2

Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

What This Lecture Covers

3.2.1

Scheduling Queues

Where do processes wait?

3.2.2

CPU Scheduling

Who runs next, and how often?

3.2.3

Context Switch

How is the CPU handed over?

Why We Need Scheduling

Five apps, one core:

Word document

YouTube video

PDF download

Python ML model

WhatsApp messages

QUESTION

How can they all appear to run together?

The OS decides which ready process gets the CPU — **process scheduling**.

Two Objectives Behind Scheduling

OBJECTIVE 1

Multiprogramming

Keep the CPU busy at all times.

Chef cooks another dish while soup simmers

OBJECTIVE 2

Time sharing

Switch often so users get quick responses.

Professor rotates among students

Multiprogramming vs. Time Sharing

	Multiprogramming	Time sharing
Goal	Keep the CPU busy	Fast, responsive service
Switches	When a process stalls	Frequently, by design
Focus	CPU utilisation	Response time
Analogy	Chef while soup cooks	Teacher rotating students

Efficiency vs. **responsiveness** — they work together.

Single-Core vs. Multicore

Single core

Processes take turns



One process at any instant.

Four cores

One process per core



Extra processes wait for a free core.

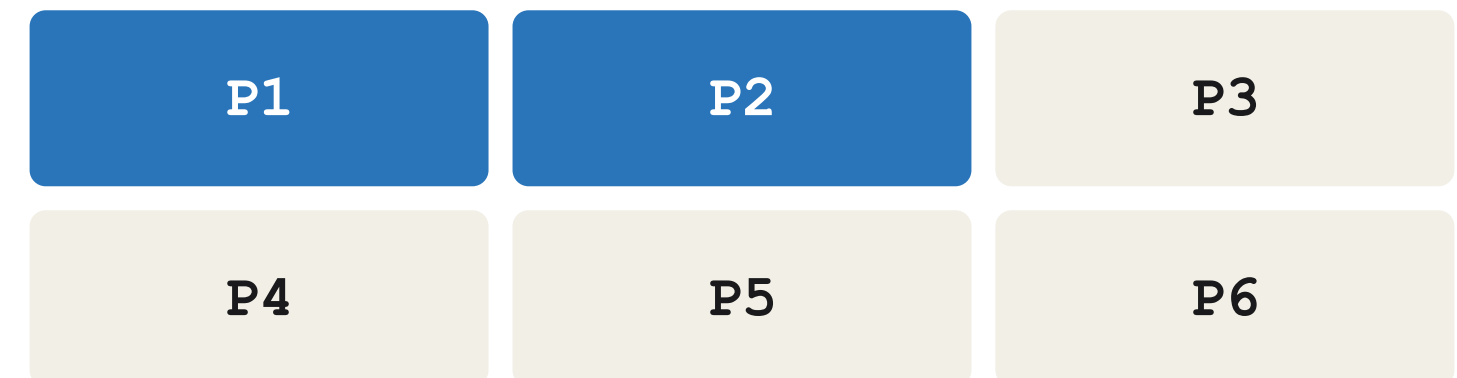
Degree of Multiprogramming

The number of processes **in memory** .

Not the number of cores.

Too many processes can **hurt** performance.

EXAMPLE · RAM



Degree = **6**, with 2 running on 2 cores.

CPU-Bound vs. I/O-Bound

CPU-bound

Mostly computing, for long periods.

ML training · matrix maths

A student solving equations

I/O-bound

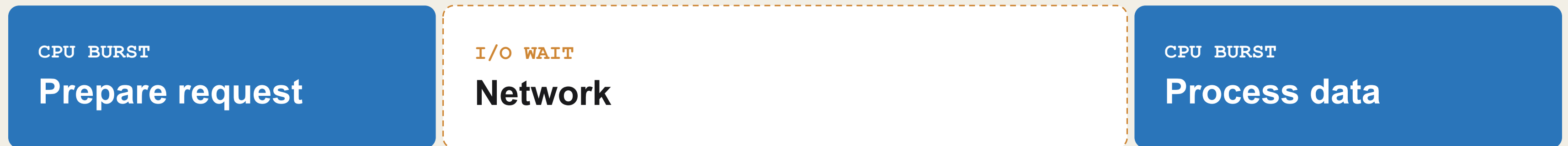
Uses the CPU briefly, then waits.

downloads · database reads

A waiter awaiting the kitchen

CPU Bursts and I/O Bursts

Example: opening a large PDF from a server



A good scheduler serves **both** kinds of workload.

3.2.1

Scheduling Queues

One ready queue for the CPU, a set of wait queues for events.

The Ready Queue

Processes ready to run, waiting for a core.

Scheduling

Choosing the next process

Dispatching

Giving it the CPU

COMPUTER

BANK

Processes

Customers

CPU core

Teller

Scheduler

Bank manager

Ready queue

Customer line

Wait Queues

Processes waiting for an **event** — disk, network, a child, a device.

GROUP A · READY QUEUE

Patients ready to see the doctor.

`needs: the doctor`

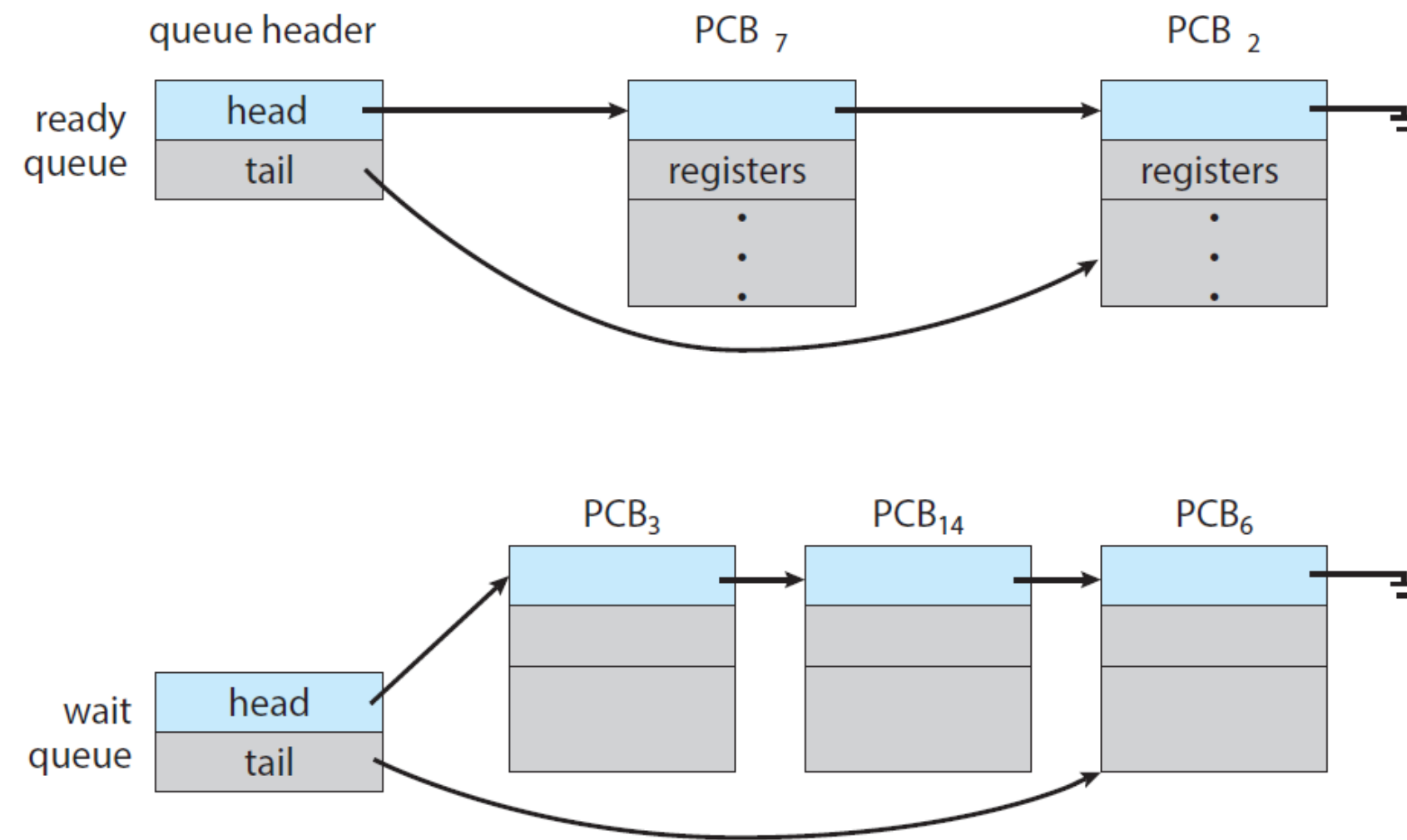
GROUP B · WAIT QUEUE

Patients awaiting lab results.

`needs: the results`

FIGURE 3.4

The Ready Queue and Wait Queues



head

Points to the first PCB

tail

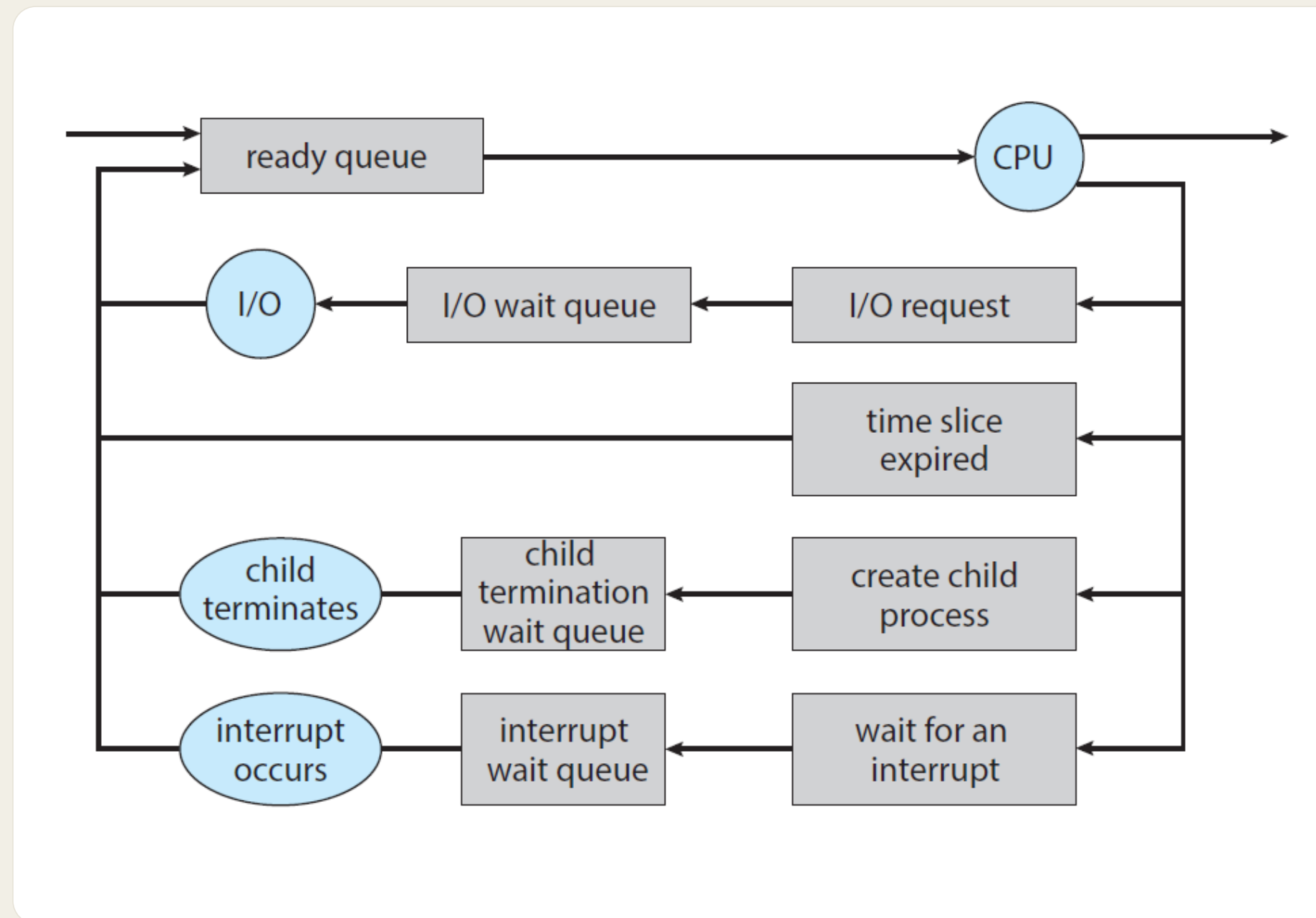
Points to the last PCB

next

Links each PCB to the next

FIGURE 3.5

Queueing Diagram of Scheduling



Circles

Resources serving the queues

Arrows

Flow of processes

Once Dispatched, What Can Happen?

case 1

Issues an I/O request

→ I/O wait queue

case 2

Time slice expires

→ back to the ready queue

case 3

Waits for a child

→ blocks until the child ends

case 4

Finishes

→ terminates, resources freed

3.2.2

CPU Scheduling

Selecting a process from the ready queue and allocating it a core.

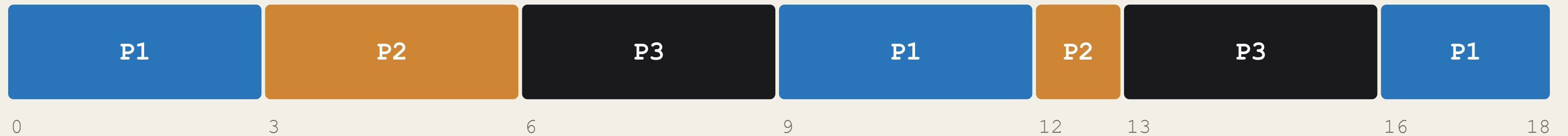
Who Runs Next?

PROCESS	ACTIVITY	CPU TIME
P1	Scientific simulation	50 ms
P2	Respond to a keystroke	3 ms
P3	Calculate student grades	20 ms

Preemption lets P2 run **before** P1 finishes.

Time Slicing with a 3 ms Quantum

Round Robin · P1 = 8 ms · P2 = 4 ms · P3 = 6 ms



13 ms

P2 completes

16 ms

P3 completes

18 ms

P1 completes

6

CPU handovers

Choosing the Quantum

Short quantum

- + More responsive
- More switching overhead

Long quantum

- + Less overhead
- Others wait longer

The OS must **balance** both.

Three Levels of Scheduler

SCHEDULER	DECIDES	HOW OFTEN
Long-term job scheduler	Which jobs enter the system	Low
Medium-term swapper	Which to suspend or resume	Occasional
Short-term CPU scheduler	Which ready process runs now	Very high

The Three Schedulers as an Airport

LONG-TERM

Immigration officer

موظف الجوازات

Who may enter at all

MEDIUM-TERM

Lounge manager

موظف صالة الانتظار

Who steps aside for now

SHORT-TERM

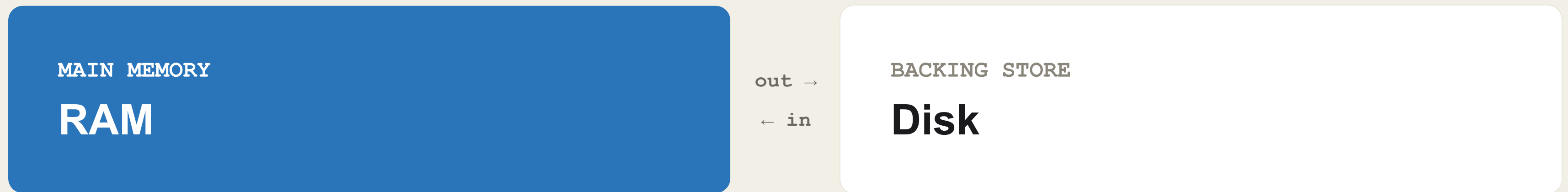
Runway controller

برج المراقبة

Who takes off next

Swapping

An intermediate form of scheduling, used when **memory is tight**.



Swap out

Save to disk, free RAM

Swap in

Restore and resume

Swapping vs. Context Switching

	Swapping	Context switch
Concern	Memory residency	CPU execution state
Moves	RAM ↔ disk	Registers ↔ PCB
Level	Medium-term	Short-term
Cost	Relatively expensive	Much faster

3.2.3

Context Switch

Save the state of one process, restore the state of another.

What “Context” Means

Everything needed to resume later — stored in the **PCB**.

CPU registers

Including the program counter

Process state

Ready, running, waiting...

Memory info

Address-space mappings

Analogy: a detailed bookmark.

Context Switch, Step by Step

1
P0 is running

2
Interrupt or block

3
Save P0 → PCB 0

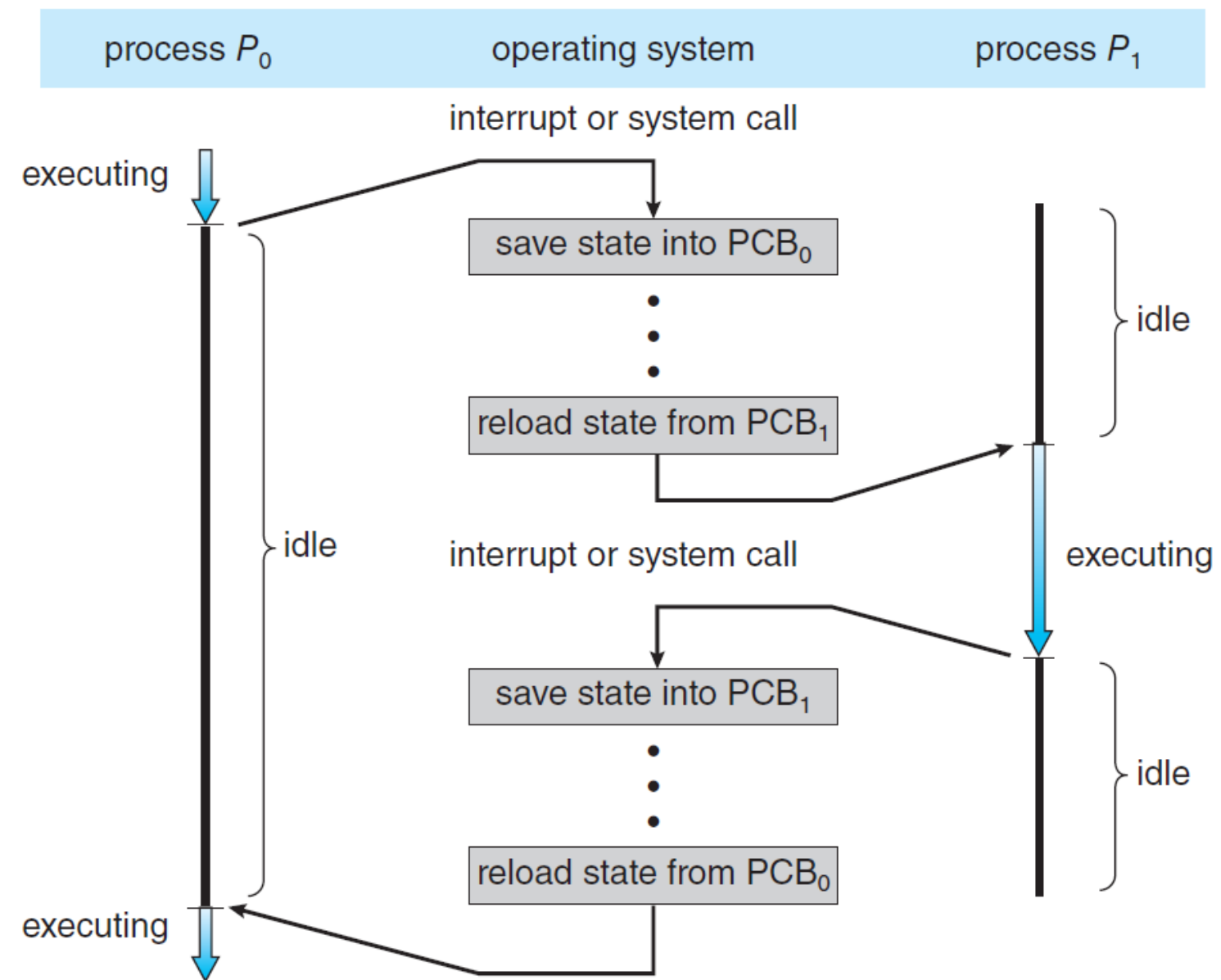
4
Scheduler picks P1

5
Restore PCB 1 → P1

6
P1 continues

FIGURE 3.6

Context Switch from Process to Process



Each process resumes — never restarts.

Not every interrupt causes a switch; the kernel may return to the same process.

Switch Time Is Pure Overhead

Cost depends on:

Memory speed

Number of registers

Hardware support

OS complexity

EXAMPLE

$$1,000 \times 5 \mu s = 5 ms$$

Time spent switching, doing no useful work.

Multitasking in Mobile Systems

iOS

Early: one **foreground** app, others suspended.

iOS 4+: limited background apps.

Android

Background work through **services**.

Music keeps playing in the browser.

Limits: **battery, heat, memory**.

Smart Hospital Scheduling

TASK	FUNCTION	WORKLOAD
P1	AI analysis of CT scans	CPU-bound
P2	Retrieve patient records	I/O-bound
P3	Generate medical reports	Mixed
P4	Receive sensor data	Often I/O-bound
P5	Respond to hospital staff	Interactive

Myth vs. Reality

MYTH

Every process in memory runs

Ready = Waiting

More processes, more speed

Swapping = context switch

Smaller quantum is always better

REALITY

Most are ready or waiting

Ready needs CPU; waiting needs an event

Excess competition slows things

Memory vs. CPU state

Too much switching wastes time

Knowledge Check

- | | | |
|---|---|----------------|
| 1 | Ready, but no CPU — where is it? | Ready queue |
| 2 | Its disk read finishes. Now what? | Becomes ready |
| 3 | Which scheduler picks who runs next? | Short-term |
| 4 | Most CPU-bound: network wait or matrices? | Matrices |
| 5 | What does a context switch do? | Save & restore |

Three Connected Ideas

CPU scheduling

Who runs next

Swapping

Who stays in memory

Context switch

How the CPU changes hands

Multiprogramming keeps the CPU **busy**; time sharing keeps users **served**.

An AI model trains while you browse, play music and download a paper.

How does the OS keep **everything** responsive?

RECAP

Key Terms from Section 3.2

multiprogramming

degree of multiprogramming

time sharing

I/O-bound

CPU-bound

ready queue

wait queue

queueing diagram

dispatch

time slice

CPU scheduler

long-term scheduler

medium-term scheduler

swapping

context switch

3.3

Operations on Processes

How processes are created and terminated.