

1.4 Operating systems

Name: _____ Class: _____ Date: _____

Total: 22 marks

KEY WORDS operating system • round robin • scheduling
• polling • interrupt

Objective

Build the skills to answer exam questions on **operating systems** – the functions of an OS, scheduling, and polling and interrupts.

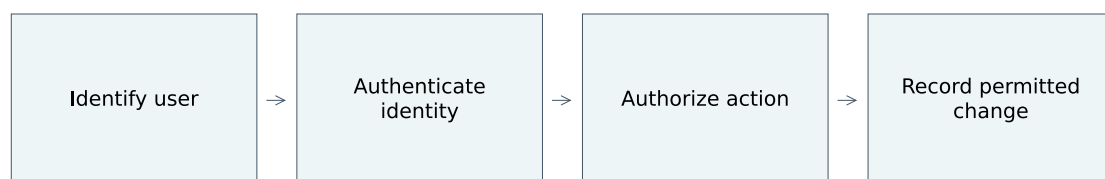
You must be able to:

- describe the functions of an **operating system**
- trace first come first served, **round robin** and priority **scheduling**
- compare **polling** with **interrupts** , and describe how an interrupt is handled

1 Key knowledge and worked examples

Study these first. Each one shows the method for a question type used later.

■ Key knowledge: what an OS does



Function	Example
process management	shares the CPU between running programs
memory management	gives each program its own protected area of RAM
file management	folders, names, read and write permissions
device management	controls hardware through drivers
security	log-in (authentication), permissions (authorization)

■ Worked example: round robin

Time slice 2 ms; P1 needs 5 ms, P2 3 ms, P3 1 ms; all arrive at 0.

Time (ms)	0–2	2–4	4–5	5–7	7–8	8–9
runs	P1	P2	P3	P1	P2	P1

1. Each process runs for at most 2 ms, then goes to the back of the queue.
2. P3 needs only 1 ms: it ends at **5**. P2 ends at **8**, P1 at **9**.
3. First come first served (P1, P2, P3) would end them at 5, 8 and **9**: P3 would wait 8 ms instead of 4.

■ Key knowledge: polling and interrupts

	Polling	Interrupt
how	CPU checks each device in turn	device signals the CPU
CPU time	wasted when nothing happens	used only when needed
suits	frequent, regular events	rare or urgent events

On an interrupt the CPU finishes the current instruction, saves its registers, runs the **interrupt handler**, restores the registers and carries on.

2 Practice

2.1 State **three** functions of an operating system.

[3]

2.2 Processes A (3 ms), B (5 ms) and C (2 ms) arrive at 0 in that order. State when each ends under (i) first come first served; (ii) round robin with a 2 ms slice. [4]

2.3 Explain why the CPU saves its registers before running an interrupt handler. [2]

3 Exam-style questions

3.1 Which OS function stops one program overwriting another's data in RAM? [1]

- ☐ **A** file management
B memory management
C device management
D the user interface

3.2 What is a weakness of priority scheduling? [1]

- ☐ **A** short jobs always wait
B low-priority processes may never run
C every process gets the same time
D it cannot run on one core

3.3 Which event is best handled by an interrupt? [1]

- ☐ **A** reading a temperature sensor every second
B a printer running out of paper
C updating a clock display every minute
D copying a file

3.4 A smartphone runs a music app, a messaging app and a game at the same time on one core.

(a) Explain how the operating system makes the three apps appear to run at the same time. [2]

(b) The game needs 6 ms, the music app 2 ms and the messaging app 3 ms. Using round robin with a 2 ms slice, in that order, state when each finishes. [3]

(c) A message arrives while the game is running. Describe how the interrupt is handled. [3]

(d) Explain why the music app should be given a higher priority than the game. [2]

Solutions

Each answer shows the steps, then the **result**.

2.1

- any three, e.g. process, memory, file or device management, security, user interface

2.2

- (i) A at **3**, B at **8**, C at **10**
- (ii) A 0–2, B 2–4, C 4–6 (ends), A 6–7 (ends), B 7–10 (ends): C **6**, A **7**, B **10**

2.3

- the handler uses the registers too, so their values would be lost
- saving them lets the interrupted program carry on exactly where it stopped

3.1 B

- each program has its own protected area

3.2 B

- starvation

3.3 B

- rare and unpredictable

3.4

(a)

- the scheduler gives each app a short time slice of the CPU in turn
- it switches so fast that all three seem to run together

(b)

- game 0–2, music 2–4 (ends), messaging 4–6, game 6–8, messaging 8–9 (ends), game 9–11 (ends)
- music **4 ms**, messaging **9 ms**
- game **11 ms**

(c)

- the CPU finishes its current instruction and saves its registers
- it runs the interrupt handler for the message
- it restores the registers and the game carries on

(d)

- music must play without gaps: a late slice is heard as a glitch
- a slightly slower game frame is less noticeable