A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

16.1 Purposes of an Operating System (OS)

Name: _____ Class: _____ Date: _____ **Total: 17 marks**

Objective

Build the skills to answer exam questions on the **operating system** 操作系统—resource and process management, and virtual memory.

You must be able to:

- explain how an OS **maximises use of resources**
- describe **process management** (states, scheduling, context switch)
- explain **virtual memory**, **paging** and **segmentation**

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

■ Managing resources

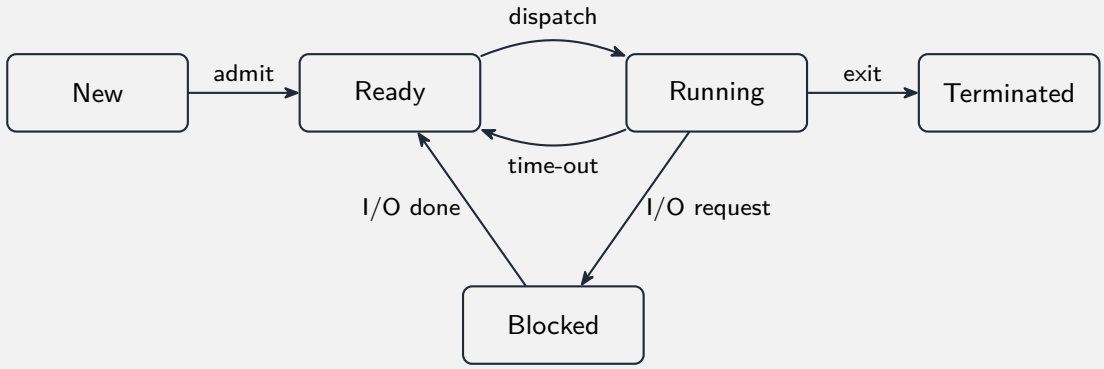
The OS shares the CPU, memory and devices fairly: **multi-tasking** (switch the CPU between processes), **memory management**, **spooling**/buffering (print jobs queue on disk), and **caching**.

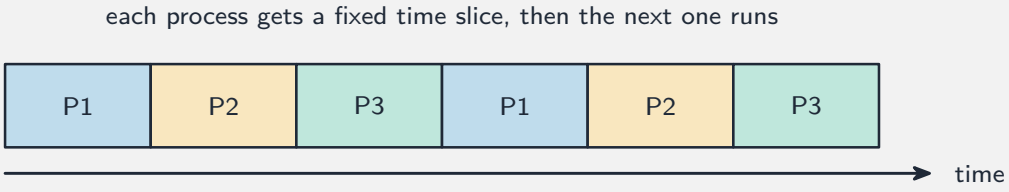
■ Process management

A **process** is a program in execution. The **scheduler** picks which ready process runs next. Common routines:

- **First Come First Served (FCFS)** —run in arrival order (simple, but a long job delays the rest).
- **Shortest Job First (SJF)** —run the job with the shortest expected time next (low average wait).
- **Shortest Remaining Time (SRT)** —pre-emptive SJF: switch to a newly-arrived shorter job.
- **Round robin** —each process gets a fixed **time slice** in turn, then goes to the back of the queue (fair, responsive).

A process moves between states:





Round-robin scheduling shares the processor fairly

Switching from one process to another saves/restores state via the **process control block** (PCB) —a **context switch**. An **interrupt** makes the CPU pause the current process; the OS **kernel** runs the matching interrupt service routine (for an I/O-complete signal, a timer, or an error), then resumes —this is how the OS responds to events.

■ Virtual memory

Each process gets its own **virtual address space**, mapped to physical RAM. In **paging**, virtual memory is split into fixed **pages**, RAM into **frames**, linked by a **page table**. A **page fault** (page not in RAM) loads it from the **swap file**, evicting another page. This lets total memory **exceed RAM**; too many faults cause **thrashing**. **Segmentation** uses variable-sized logical segments (code, stack, heap).

2 Practice

Now apply the methods above.

2.1 State **two** resources that an operating system manages. [1]

2.2 State what **round-robin** scheduling does. [1]

2.3 Name the **five** process states. [2]

2.4 State what a **context switch** is. [1]

3 Exam-style questions

3.1 Draw the **process state diagram** with the states New, Ready, Running, Blocked and Terminated, and label the transitions. [4]

3.2 Explain what happens to a **running** process when it requests **input/output**. [2]

3.3 Explain how **virtual memory** lets programs use more memory than the physical RAM. [3]

3.4 State what causes **thrashing**. [1]

3.5 Describe the **Shortest Job First (SJF)** scheduling routine and give **one** benefit of it. [2]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **16.1 Purposes of an Operating System** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/32 J25 —Q5 (process states and scheduling)
- 9618/31 N25 —Q4 (scheduling routines)
- 9618/33 J25 —Q6 (the kernel and interrupts)
- 9618/33 N24 —Q8 (memory management)

Solutions

2.1 Any two of: CPU (time), memory/RAM, disk/storage, I/O devices.

2.2 Gives each process a **fixed time slice** of CPU in turn, then moves it to the **back of the queue**.

2.3 New, Ready, Running, Blocked, Terminated.

2.4 Saving the state of one process (to its PCB) and **restoring** another's, so the CPU can switch between them.

3.1 Correct states and transitions: New→Ready (admit); Ready Running (dispatch / time-out); Running→Blocked (I/O request) and Blocked→Ready (I/O done); Running→Terminated (exit).

3.2 The process moves from **Running** to **Blocked**; the scheduler then **dispatches another ready process** to the CPU while it waits.

3.3 Each process has a **virtual address space** split into **pages**; only the pages in use are kept in RAM **frames**, the rest stay in the **swap file** on disk; on a **page fault** the OS swaps a page in (evicting one if needed), so the program can be larger than physical RAM.

3.4 Too little RAM for the active pages, so the OS spends most of its time **swapping pages in and out** instead of doing useful work.

3.5 SJF runs the ready process with the **shortest expected run time** next; benefit: the **lowest average waiting time** —short jobs are not stuck behind long ones.