A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

5.1 Operating Systems

Name: _____ Class: _____ Date: _____ **Total: 31 marks**

KEY WORDS operating system 操作系统 • program library 程序库 • process 进程 • memory management 内存管理 • process management 进程管理

Objective

Build the skills to answer exam questions on the **operating system** 操作系统 — why it is needed, its management tasks, utility software and program libraries.

You must be able to:

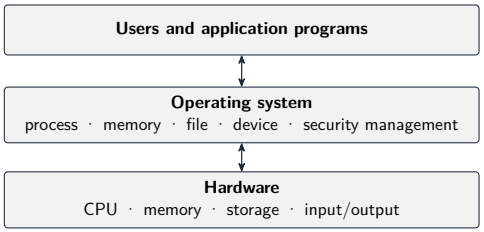
- explain **why** a computer needs an operating system
- describe the key **management tasks** the OS performs
- describe typical **utility software** and the use of **program libraries**

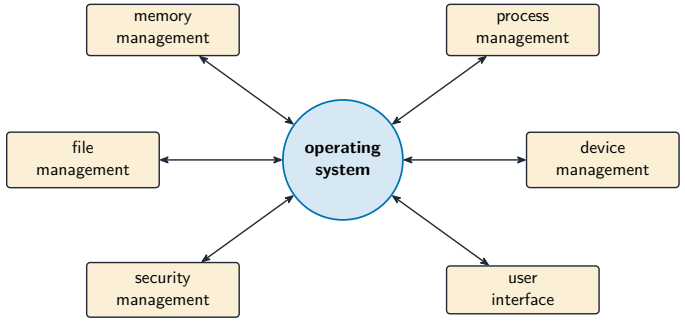
1 Worked examples

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.

■ Why an OS, and where it sits

Hardware alone only fetches and runs instructions. The **operating system** is the software layer that manages the hardware and provides services, so programs do not talk to the hardware directly:





The core tasks an operating system manages

■ The management tasks

Task	What it does
process management	load programs, schedule CPU time, switch between processes
memory management	give each process memory, keep them apart, use virtual memory
file management	organise files/folders, control permissions
device management	handle I/O through device drivers, buffer data
security management	user accounts, permissions, encryption

It also provides the **user interface** and **networking**.

■ Utility software and libraries

Utility programs maintain the system: disk/file management, **defragmenter** (reorders fragmented files on a hard disk), backup, antivirus, compression. A **program library** is a collection of ready-made, tested routines a programmer can reuse instead of rewriting them.

2 Practice

Now apply the methods above.

2.1 State **two** reasons a computer needs an operating system. [2]

2.2 Name the OS task that gives each running program CPU time and switches between programs. [1]

2.3 State what **memory management** does. [2]

2.4 State one benefit of using a **program library**. [1]

3 Exam-style questions

3.1 Explain why application programs do not access the hardware directly. [3]

3.2 Describe **two** management tasks carried out by an operating system. [4]

3.3 (a) State what a **disk defragmenter** does. [2]

(b) State why a defragmenter is **not** useful on a solid-state drive (SSD). [1]

3.4 Explain how **memory management** keeps two running programs from interfering with each other. [2]

3.5 An operating system manages a shared school computer.

(a) **State** two tasks performed by **hardware management**. [2]

(b) **State** two tasks performed by **security management**. [2]

(c) **Describe** how the operating system's **file management** lets two students save work with the same file name without either losing it. [3]

(d) A programmer building a new attendance system uses **library routines** for the date handling and the encryption. **Describe** one benefit and one drawback of doing so. [4]

(e) **Explain** why a **utility program** such as a disk defragmenter is not part of the operating system kernel itself. [2]

Solutions

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

2.1

- any two of: manages the hardware for programs; lets several programs share the hardware safely; provides services (files, network); provides a user interface

2.2

- **process** (or CPU) management

2.3

- gives each process the memory it needs and **keeps processes apart**
- uses virtual memory so the working set can exceed physical RAM

2.4

- it provides ready-made, tested routines, so the programmer **saves time** and avoids re-writing/re-testing code

3.1

- the hardware is complex and shared; if every program drove it directly they could clash or corrupt each other
- the OS provides a **safe, uniform interface** and shares the hardware fairly
- so programs are simpler and protected from one another

3.2

- any two, each described, e.g. **process management** — schedules CPU time and switches between processes
- **memory management** — allocates memory and keeps processes apart

3.3

(a) it moves the scattered pieces of fragmented files so each file's blocks are **together**

- that reduces seek time

(b) an SSD has no moving parts / no seek time, so reordering blocks gives no benefit (and wears the drive)

3.4

- the OS gives each program its **own block of memory** and checks every access
- an access outside a program's block is blocked, so one program cannot read or change another's memory

3.5

(a) any two: install and use **device drivers** so peripherals work; allocate devices to processes and queue their requests; manage the buffers between fast processor and slow

devices; handle interrupts from hardware

(b) any two: authenticate users with usernames and passwords; set and enforce **access rights** on files; keep an audit log of who did what; run or schedule backups; apply security updates

(c) the file system keeps a **directory structure**, and each student's work is stored in a different directory

- a file is identified by its **full path**, not just its name, so two identical names in different directories are different files
- the OS also holds each user's access rights, so one student cannot overwrite the other's file even by accident

(d) benefit: the routines are already written and **tested** by others, so development is faster and the code is likely to have fewer faults — encryption in particular is very hard to write correctly

- drawback: the programmer does not control the code and may not understand it fully, so a fault or a security weakness in the library becomes a fault in their program, and the library may be larger than needed or stop being maintained

(e) the kernel holds only what must run with full privilege and be available at all times

- a defragmenter is an occasional maintenance task that can run as an ordinary program, and keeping it out of the kernel keeps the kernel smaller and less likely to fail