Week 12

A-Level Computer Science

Homework bundle for this week

本周作业合集

exercises.lol

Contents 目录

Topic 5 quiz 2

Exercise sheet 5.1 5

Past-paper questions 5.1 11

Exercise sheet 5.2 15

Past-paper questions 5.2 21

5. System Software

Starter Quiz · 课前小测

A-Level Computer Science Name: _____ Score: _____

1. Which is a core job of an operating system?

   ☐ managing the hardware so programs can share it safely

   ☐ designing the CPU chip

   ☐ writing the user's documents

   ☐ manufacturing the hardware

2. A program library is:

   ☐ pre-written code (subroutines, classes) that programs reuse

   ☐ a folder of the user's documents

   ☐ the operating system kernel

   ☐ a kind of CPU register

3. A compiler differs from an interpreter because it:

   ☐ translates the whole program once and produces a stand-alone executable

   ☐ runs the program one line at a time

   ☐ must be installed to run the finished program

   ☐ never reports errors

4. The operating system lets several programs share the same hardware safely, so they don't interfere with each other.

   ☐ True ☐ False

5. A benefit of using a library is that the code is:

   ☐ well-tested and reliable, saving development time

   ☐ always smaller than writing it yourself

   ☐ impossible to update

   ☐ specific to one program only

6. When does an interpreter report an error?

   ☐ when it reaches the line containing the error

   ☐ only after the whole program is translated

   ☐ never

   ☐ before the program starts

7. A dynamic library can be shared by many programs and updated once for all of them.

   ☐ True ☐ False

8. A collection of ready-made, reusable, tested code is a program _____.

   _____

9. System-maintenance programs such as antivirus, backup and defragmenter are called _____ software.

   _____

10. An interpreter produces no stand-alone executable, so the interpreter itself must be installed to run the program.

    ☐ True ☐ False

A-Level Computer Science

1. **managing the hardware so programs can share it safely**

   The OS manages hardware and provides services and a UI, letting multiple programs run safely on shared hardware.

2. **pre-written code (subroutines, classes) that programs reuse**

   Libraries provide ready-made, tested code (maths, graphics, network…) that a program can call instead of writing its own.

3. **translates the whole program once and produces a stand-alone executable**

   A compiler translates all at once into an executable that runs without the compiler. An interpreter runs line by line.

4. **True**

   Sharing the CPU, memory and devices safely between many programs is exactly what the OS is for.

5. **well-tested and reliable, saving development time**

   Library code is tested, widely used and standardised — reusing it saves time and improves reliability.

6. **when it reaches the line containing the error**

   An interpreter runs line by line, so it reports an error at the moment it reaches that line — handy during development.

7. **True**

   Because it is loaded at run time, one shared dynamic library serves many programs; updating it fixes them all without rebuilding each.

8. **library**

   Libraries save time and reduce bugs.

9. **utility**

   Utility software maintains the system; word processors and browsers are application software.

10. **True**

    An interpreter translates and runs line by line each time — there is no separate executable to ship, unlike a compiler.

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:

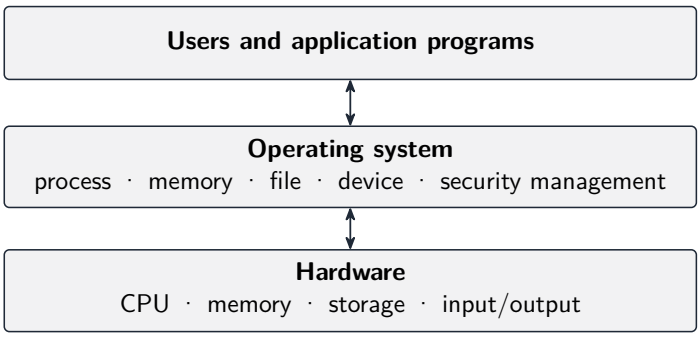
- explain **why** a computer needs an operating system
- describe the key **management tasks** the OS performs
- describe typical **utility software** and the use of **program libraries**

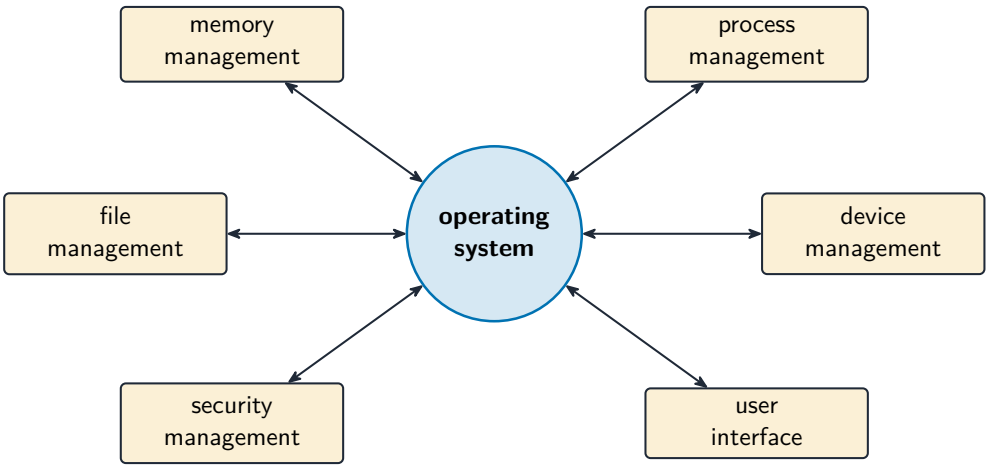
1 Worked examples

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

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**, reducing 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 operating system 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.

5 The Operating System (OS) is responsible for hardware and security management in a computer system.

(a) (i) State **two** tasks that are performed by hardware management.

1 .....

.....

2 .....

.....

[2]

(ii) State **two** tasks that are performed by security management.

1 .....

.....

2 .....

.....

[2]

(b) (i) Describe **one** benefit of using library routines when developing software.

.....

.....

.....

..... [2]

(ii) Describe **one** drawback of using library routines when developing software.

.....

.....

.....

..... [2]

(c) Identify **two** presentation features found in a typical Integrated Development Environment (IDE).

1 .....

2 .....

[2]

4 A student uses a laptop to write a program that is saved as a text file.

(a) The laptop has utility software and an Operating System (OS).

(i) Describe the file management tasks carried out by an OS.

.....

.....

.....

..... [2]

(ii) Explain the need for back-up software.

.....

.....

.....

..... [2]

(b) The student compresses the file before it is emailed to their teacher as an attachment.

(i) Explain the benefits to the teacher of the attachment being a compressed file.

.....

.....

.....

.....

.....

..... [3]

(ii) Describe **one** lossless method of compressing a text file.

.....

.....

.....

.....

.....

..... [3]

(c) The student used a program library when writing their program.

Explain the benefits to the student of using library files when writing a program.

.....

.....

.....

.....

.....

..... [3]

(d) The program code is written using an Integrated Development Environment (IDE).

(i) One presentation feature found in a typical IDE is prettyprint.

Identify **and** describe **one other** presentation feature found in a typical IDE.

Feature .....

Description .....

.....

.....

[2]

(ii) One debugging feature found in a typical IDE is single stepping.

Identify **and** describe **one other** debugging feature found in a typical IDE.

Feature .....

Description .....

.....

.....

[2]

6 A computer has an Operating System (OS).

Memory management and process management are two OS tasks.

Explain how memory management **and** process management support multi-tasking.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

5.2 Language Translators

Name: _____ Class: _____ Date: _____ **Total: 30 marks**

KEY WORDS interpreter 解释器 • compiler 编译器 • integrated development environment (ide) 集成开发环境 • assembler 汇编器 • assembly language 汇编语言

Objective

Build the skills to answer exam questions on **language translators** 翻译器 — assemblers, compilers and interpreters, when to use each, and the features of an IDE.

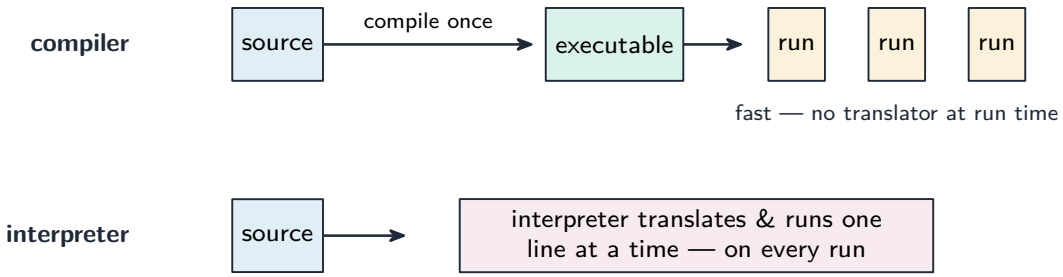
You must be able to:

- explain the need for an **assembler**, a **compiler** 编译器 and an **interpreter** 解释器
- compare and justify the use of a compiler or an interpreter
- explain the **hybrid** (bytecode) approach used by Java
- describe features of an **Integrated Development Environment (IDE)** 集成开发环境

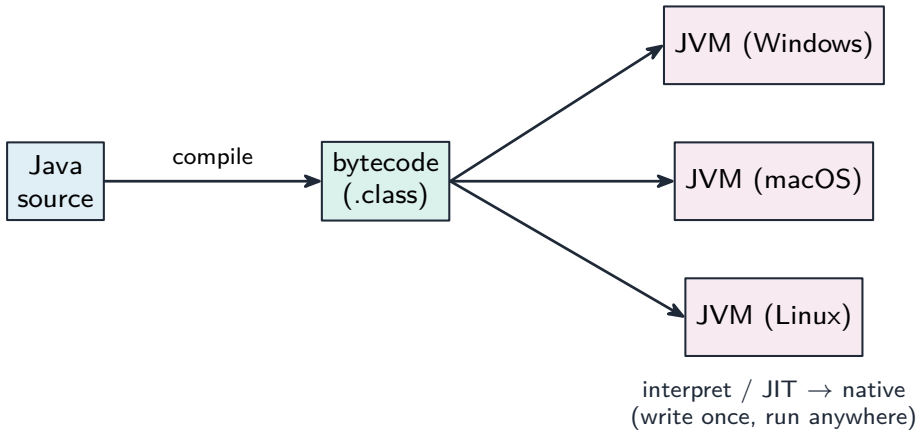
1 Worked examples

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.

■ Three translators



Compiler: translate all at once, then run. Interpreter: translate and run line by line.



Java compiles to bytecode run by a virtual machine

You write source code; the CPU runs **machine code** 机器码. A translator converts between them.

- **assembler** — translates **assembly language** to machine code (one-to-one).
- **compiler** — translates a whole **high-level** program to machine code **once, before it runs**.
- **interpreter** — translates and runs a high-level program **one line at a time**.

Compiler vs interpreter

	compiler	**interpreter**
When translated	all at once, before running	line by line, while running
Errors	reports **all** at compile time	stops at the **first** error reached
Output	a stand-alone **executable**	none — re-translates each run
Needs the translator to run?	no	yes
Run-time speed	faster	slower
Portable across platforms?	recompile per platform	yes (if interpreter exists)

Choosing: use a **compiler** for speed and to distribute finished software; use an **interpreter** for quick development and cross-platform scripts.

Java (hybrid) and IDEs

Java is **compiled to bytecode** 字节码, which a **virtual machine** (the JVM) interprets (or just-in-time compiles). So errors are caught early **and** the bytecode runs anywhere with a JVM ("write once, run anywhere"). An **IDE** brings the tools together, with features that help at each stage:

- **writing code:** an editor with **syntax highlighting**, **auto-complete** / context-sensitive prompts, and **prettyprinting** (auto-indenting).
- **finding errors: error diagnostics** that point to the line of a syntax error, and reports as you type.
- **debugging:** a **debugger** with **breakpoints**, **single-stepping**, and a **watch**

window to inspect variable values while paused.

2 Practice

Now apply the methods above.

2.1 State the difference between a **compiler** and an **interpreter**. [2]

2.2 Give one situation in which an **interpreter** is more suitable than a compiler. [1]

2.3 State what an **assembler** translates. [1]

2.4 Name **two** features of an IDE. [2]

3 Exam-style questions

3.1 (a) State **two** benefits of using a **compiler** rather than an interpreter. [2]

(b) State **two** benefits of using an **interpreter** rather than a compiler. [2]

3.2 Explain how Java's "compile to bytecode, then run on a virtual machine" approach lets the same program run on different computers. [3]

3.3 Describe **two** features an IDE provides to help a programmer **debug** a program. [2]

3.4 A company sells finished software to customers who do not have any programming tools installed. State which translator the company should use, and justify your choice.[2]

3.5 A programmer writes a program in a high-level language and releases it as **free software**.

(a) **Explain** how the programmer can use an **interpreter** first and then a **compiler** while developing the same program. [4]

(b) **State** two things an **Integrated Development Environment** provides beyond a translator. [2]

(c) A user downloads the program. **Describe** what "free software" allows the user to do that **freeware** does not. [3]

(d) **Explain** why the program still runs more slowly when interpreted than when compiled, even on the same computer. [3]

(e) The compiler reports a **syntax error** on a line that looks correct. **Suggest** one reason this can happen. [1]

Solutions

2.1 A compiler translates the **whole** program **once** into an executable; an interpreter translates and runs it **one line at a time**.

2.2 e.g. during development (quick edit–run cycles) / for a cross-platform script / a small or run-once program.

2.3 Assembly language (into machine code).

2.4 Any two of: syntax highlighting; auto-complete; one-key compile/run; debugger; version-control integration.

3.1 (a) Any two: runs faster (no translation at run time); produces a stand-alone executable; the translator is not needed to run it; the source code is not given away.
(b) Any two: reports errors as it reaches them (easier development); no need to recompile after each change; the same program runs on any platform with the interpreter.

3.2 The program is compiled once into **bytecode**, which is platform-independent; any computer with a **JVM** can interpret/run that bytecode; so it runs unchanged on different operating systems — "write once, run anywhere".

3.3 Any two: set **breakpoints** to pause execution; **step through** line by line; **inspect variable** values while paused.

3.4 A **compiler** — it produces a stand-alone executable that runs without any programming tools installed on the customer's computer.

3.5 (a) During development an **interpreter** translates and runs one statement at a time, so the programmer sees the effect of a change immediately and can stop at the first error. Once the program works, a **compiler** translates the whole source into machine code once, producing an executable that runs faster and can be distributed without the source.
(b) Any two: an editor with syntax highlighting and auto-complete; a debugger with breakpoints, single-stepping and variable watches; a project manager for multiple source files; built-in help and error reporting.
(c) Free software gives the user the source code and the right to **study**, **modify** and **redistribute** it, including modified versions. Freeware costs nothing to use but the source is not provided and the licence forbids changing or redistributing it.
(d) An interpreter must **translate each statement every time it is executed**, so a statement inside a loop is translated on every pass. Compiled code was translated once, before running, so the processor executes machine code directly with no translation overhead at all.
(e) The real error is on an **earlier** line — for example a missing closing bracket or quotation mark — and the compiler only notices when the statement fails to make sense.

10 A programmer is developing a computer program.

(a) Explain how the programmer can first use an interpreter **and** then a compiler to develop the computer program.

Interpreter .....

.....

.....

.....

Compiler .....

.....

.....

.....

[4]

(b) The programmer releases the program as Free Software.

Describe what is meant by Free Software.

.....

.....

.....

..... [2]

(c) A user downloads the computer program from the internet.

State what should be included as part of the download to make sure the program is authentic.

..... [1]

8 Compilers and interpreters are used to translate programs written in a high-level language into a low-level language.

(a) State **two** disadvantages of using a compiler compared to an interpreter during program development.

1 .....

.....

2 .....

.....
[2]

(b) Explain how a programmer benefits from using program libraries during program development.

.....

.....

.....

.....

.....

..... [3]

(c) A programmer is developing an Operating System (OS).

Identify **two** types of software licence the programmer can use to allow other people to edit and redistribute the OS.

1 .....

2 .....
[2]

(d) Memory management is one key management task performed by an OS.

Give **three** other key management tasks that are performed by an OS.

1 .....

2 .....

3 .....
[3]

3 A programmer is writing a program in a high-level language.

(a) Complete the description of compilers and interpreters by writing the missing words.

A compiler checks all the code before attempting to translate the program. If any errors are found, they are all reported at the same time and the program does not translate or run. If there are no errors found, the compiler produces ..... which can run without access to the ..... .

An interpreter translates one line of code and then runs it, before moving to the next line of code. If the line of code has an error, the interpreter ..... and displays the error. The programmer can correct the error ..... and then the interpreter continues translating from that point.

[4]

(b) The programmer needs to keep the program files secure on their computer and during electronic data transmission over the internet. The files are protected by a password.

Complete the table by identifying **one** other method of keeping the files secure during electronic data transmission **and one** other method of keeping the files secure on the computer.

State how each method protects the data.

The methods must be different.

	Method	**How the method protects the data**
during transmission	.....	.....
on the computer	.....	.....

[4]

(c) The computer program allows users to play a game in a virtual world. Users can play the game from any computer with internet access using a web browser. The game allows users to interact with other users.

Explain the reasons why the statement 'This computer game uses a client-server model' is correct.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(d) Describe the possible consequences of the programmer **not** joining a professional ethical body.

.....

.....

.....

.....

.....

..... [3]

4 A program is written in a high-level language by a team of three programmers using an Integrated Development Environment (IDE).

(a) Describe how the programmers can use the debugging features of a typical IDE during the development of the program.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) The programmers created a new program library whilst developing the program.

Describe the benefits to the programmers of creating a program library.

.....

.....

.....

.....

.....

..... [3]

(c) The file containing the final program code will be sent by email for beta testing.

Identify **one** security method that can be used to protect the program code from unauthorised access during email transfer.

Explain how your chosen method protects the program code.

Security method .....

.....

Explanation .....

.....

.....

.....

[3]

3 A software developer is writing a computer program.

(a) The developer uses an interpreter while writing the program code because it is easier for debugging.

Explain **one** reason why it is easier to debug the program code using an interpreter instead of a compiler.

.....

.....

.....

..... [2]

(b) The program is ready to be sold to customers.

The developer uses a compiler because it creates an executable file.

Explain the reasons why the need to create an executable file makes the complier the appropriate choice when the program is complete.

.....

.....

.....

.....

.....

..... [3]

8 A programmer uses an Integrated Development Environment (IDE) to write a computer program. The IDE has both a compiler and an interpreter as built-in translators.

(a) The programmer decides to use the compiler when testing the final program.

Describe the benefits of using the compiler during testing.

.....

.....

.....

..... [2]

(b) IDEs have many features other than built-in translators.

Complete the table by identifying **one other** common IDE feature that can be used for each purpose. Describe how each feature helps the user during program development.

Each feature must be different. Do **not** give translator as one of your features.

Purpose	IDE feature	Description
for coding	....	....
for presentation	....	....
for debugging	....	....

[6]

(c) The programmer uses program libraries when developing the program.

Describe **two** benefits to the programmer of using program libraries.

1 .....

.....

2 .....

.....

[2]