

Exercise walk-through

16.1

Purposes of an Operating System (OS)

A-Level Computer Science

The user interface hides the hardware

The user sees windows, menus and folders – not addresses or sectors. One click on an icon makes the OS find the program, allocate memory, load it and start it.

CLI

powerful, scriptable

GUI

easier to learn

Most systems offer both. The interface is an abstraction, not a different OS.

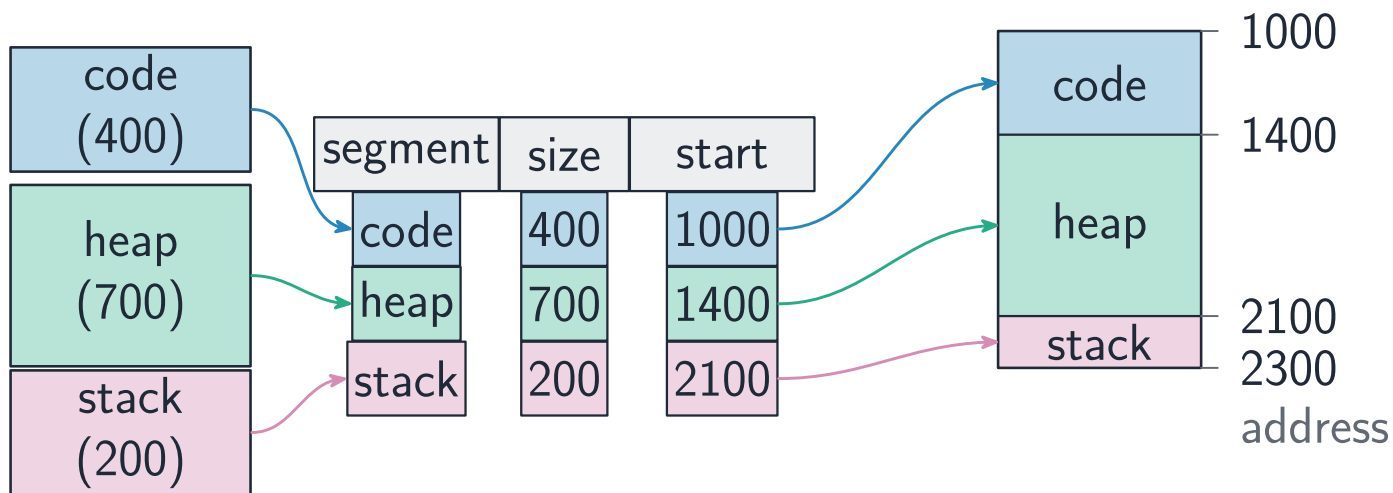
Paging and segmentation

Paging splits memory into equal **pages** ; segmentation splits it into logical, unequal **segments** .

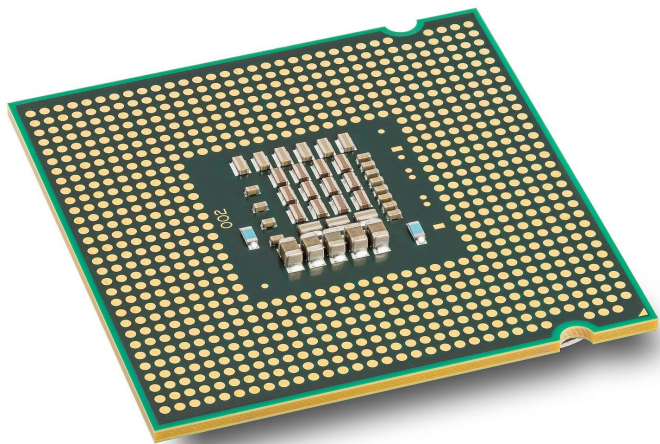
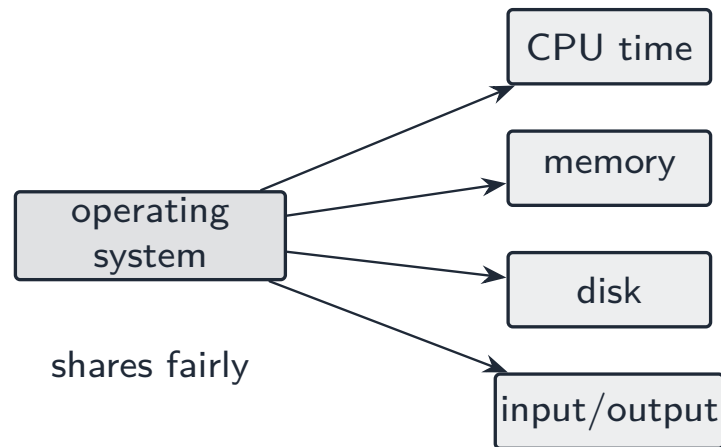
logical segments
(one process)

segment table

physical memory



How an OS maximises use of resources



*The OS shares the CPU, memory, disk and I/O
between programs*

The processor is a key resource the OS shares between competing tasks

Exercises 2.1-2.3

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]

Exercises 2.1-2.3

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

2.1

- any two of: CPU (time), memory/RAM, disk/storage, I/O devices
-

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

2.2

- gives each process a **fixed time slice** of CPU in turn, then moves it to the **back of the queue**
-

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

2.3

- New, Ready, Running, Blocked, Terminated

Exercises 2.4-2.6

2.4 State what a **context switch** is.

[1]

2.5 Identify **two** scheduling routines.

[2]

2.6 State **one** difference between **paging** and **segmentation**.

[1]

Exercises 2.4-2.6

2.4 State what a **context switch** is.

[1]

2.4

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

2.5 Identify **two** scheduling routines.

[2]

2.5 any two:

- first come first served (FCFS); shortest job first (SJF); shortest remaining time (SRT); round robin

2.6 State **one** difference between **paging** and **segmentation**.

[1]

2.6 any one:

- paging uses **fixed-size** pages; segmentation uses **variable-size** segments
- pages are **physical** divisions; segments are **logical** units of the program

Exercises 2.7-3.1

2.7 Two processes arrive at time 0: A needs **5 ms** and B needs **1 ms**. Calculate the **average waiting time** if they run **A then B**, and if they run **B then A**. [2]

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

Exercises 2.7-3.1

2.7 Two processes arrive at time 0: A needs **5 ms** and B needs **1 ms**. Calculate the **average waiting time** if they run **A then B**, and if they run **B then A**. [2]

2.7

- A then B: A waits 0, B waits 5; average = $\frac{0 + 5}{2} = \mathbf{2.5 \text{ ms}}$
- B then A: B waits 0, A waits 1; average = $\frac{0 + 1}{2} = \mathbf{0.5 \text{ ms}}$

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

3.1

- New $\rightarrow$ Ready (admit)
- Ready $\rightleftharpoons$ Running (dispatch / time-out)
- Running $\rightarrow$ Blocked (I/O request) and Blocked $\rightarrow$ Ready (I/O done)
- Running $\rightarrow$ Terminated (exit)

Exercises 3.2-3.3

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]

Exercises 3.2-3.3

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

3.2

- the process moves from **Running** to **Blocked**
- the scheduler then **dispatches another ready process** to the CPU while it waits

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

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

Exercises 3.4-3.5

3.4 State what causes **thrashing**.

[1]

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

[2]

Exercises 3.4-3.5

3.4 State what causes **thrashing**.

[1]

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

[2]

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

Exercises 3.6-3.7

3.6 Describe **two** ways in which the complexities of the computer hardware are **hidden** from the user. [2]

3.7 Explain the function of the **round robin** scheduling routine and give a benefit of this routine. [3]

Exercises 3.6-3.7

3.6 Describe **two** ways in which the complexities of the computer hardware are **hidden** from the user. [2]

3.6 any two:

- a **user interface** (GUI) lets the user work without touching the hardware
- **device drivers** let programs use devices without knowing how they work
- **system calls** / **an API** carry out hardware tasks for programs
- **file management** shows named files and folders, not physical storage locations

3.7 Explain the function of the **round robin** scheduling routine and give a benefit of this routine. [3]

3.7

- each process gets a **fixed time slice** of the processor, **in turn**
- when its slice ends, the process is **pre-empted** and goes to the **back of the ready queue**
- benefit: it is **fair** – every process gets processor time / no process **starves** / good **response times**

Exercise 3.8

3.8 (a) Describe the process of **segmentation** for memory management. [3]

(b) Explain what is meant by **disk thrashing**. [2]

Exercise 3.8

3.8 (a) Describe the process of **segmentation** for memory management. [3]

(b) Explain what is meant by **disk thrashing**. [2]

3.8

(a) any three:

- memory is divided into **variable-size** blocks called **segments**
- each segment is a **logical unit** of the program, e.g. a procedure or a data block

Full solution on next slide

Full solution 3.8

3.8

(a) any three:

- memory is divided into **variable-size** blocks called **segments**
- each segment is a **logical unit** of the program, e.g. a procedure or a data block
- a **segment table** stores each segment's **start address** and **length**
- an address is given as a **segment number** and an **offset** within the segment

(b)

- pages are **swapped** between RAM and disk **so often**
- that the processor spends more time **transferring pages** than running the program, so performance **collapses**

Exercise 3.9 (a)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(a) Calculate the **average waiting time** using **FCFS**.

[2]

Exercise 3.9 (a)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(a) Calculate the **average waiting time** using **FCFS**.

[2]

(a)

- FCFS waits: W 0, X 8, Y 12, Z 21
- average = $\frac{0 + 8 + 12 + 21}{4} = \mathbf{10.25 \text{ ms}}$

Exercise 3.9 (b)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(b) Calculate the **average waiting time** using **SJF**.

[2]

Exercise 3.9 (b)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(b) Calculate the **average waiting time** using **SJF**.

[2]

(b)

- SJF order X, Z, W, Y; waits: X 0, Z 4, W 9, Y 17
- $\text{average} = \frac{0 + 4 + 9 + 17}{4} = \mathbf{7.5 \text{ ms}}$

Exercise 3.9 (c)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(c) Outline how the **kernel** acts as an **interrupt handler** when the time slice of a process ends. [3]

Exercise 3.9 (c)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(c) Outline how the **kernel** acts as an **interrupt handler** when the time slice of a process ends. [3]

(c) any three:

- the **timer** sends an **interrupt** when the slice ends
- the kernel **saves the state** (registers, PC) of the running process in its PCB
- it runs the **ISR** for the timer interrupt, found from the **interrupt vector table**
- the **scheduler** moves the process to the ready queue and **dispatches** the next one

Exercise 3.9 (d)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(d) Explain why a system that uses **round robin** may still be chosen, even though SJF gives a lower average waiting time. [2]

Exercise 3.9 (d)

3.9 An operating system runs four processes, which all arrive at time 0 in the order W, X, Y, Z.

Process	W	X	Y	Z
Burst time (ms)	8	4	9	5

(d) Explain why a system that uses **round robin** may still be chosen, even though SJF gives a lower average waiting time. [2]

(d)

- SJF needs each burst time **in advance**, which is usually not known
- round robin is **fair**: long processes are **not starved**, and every process **responds** quickly