From last time

Does each of the following appear in processes, programs, both, or neither?

– instructions
– read-only data
– registers
– a stack
– a heap
– network connections
– system calls
– a shared data area

MANCHESTER 1824

The University of Manchester

COMP25111: Operating Systems

Lecture 6: An Introduction to Process (and Thread) Scheduling

John Gurd

School of Computer Science, University of Manchester

Autumn 2015

Overview & Learning Outcomes

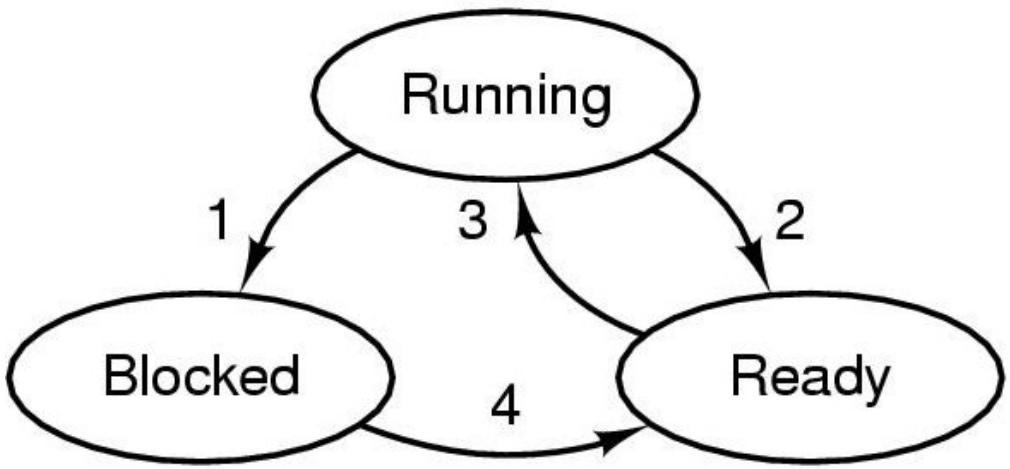
Scheduling

– When to change process

– How to select the next process to run

– Criteria

Process States & Transitions



(OSC/J fig3.2 (older - fig4.1); MOS fig2-2)

Basic Concepts

Scheduler:
– component of OS process manager
– decides which “ready” process to run next
  1 per CPU (core)
– “scheduling algorithm”

– processes **or** kernel-level threads

– CPU time was expensive, so scheduling very important
– PC (1 user & cheap) but scheduling ever more sophisticated

When to schedule?

When a process frees the CPU

When a new process joins the “ready” list

CPU burst: executing on CPU
I/O burst: blocked, waiting for I/O

Process alternates between CPU & I/O bursts

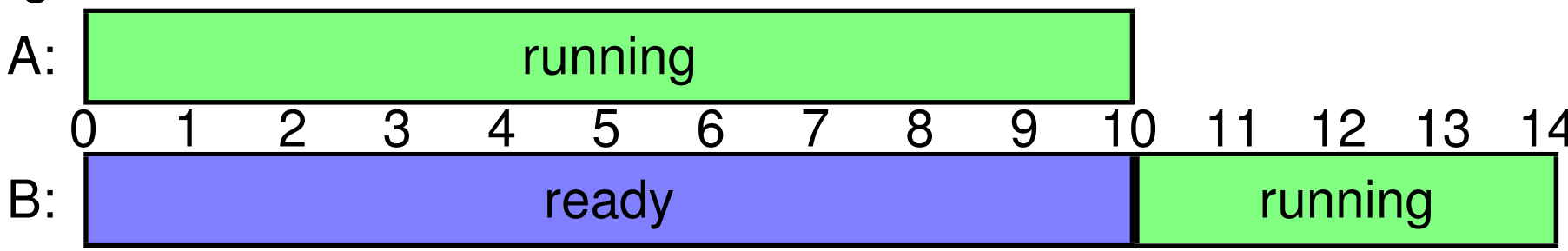
CPU bound: long CPU bursts
I/O bound: short CPU bursts

A very long CPU burst keeps other processes waiting

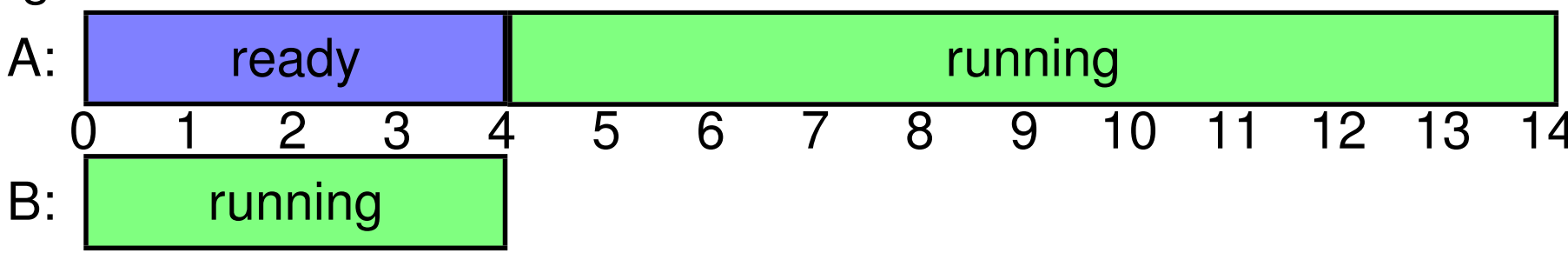
Example

Two processes, A and B, arrive at time 0;
CPU-burst time length: A=10, B=4 time-units (total=14)

e.g. run A then B



e.g. run B then A



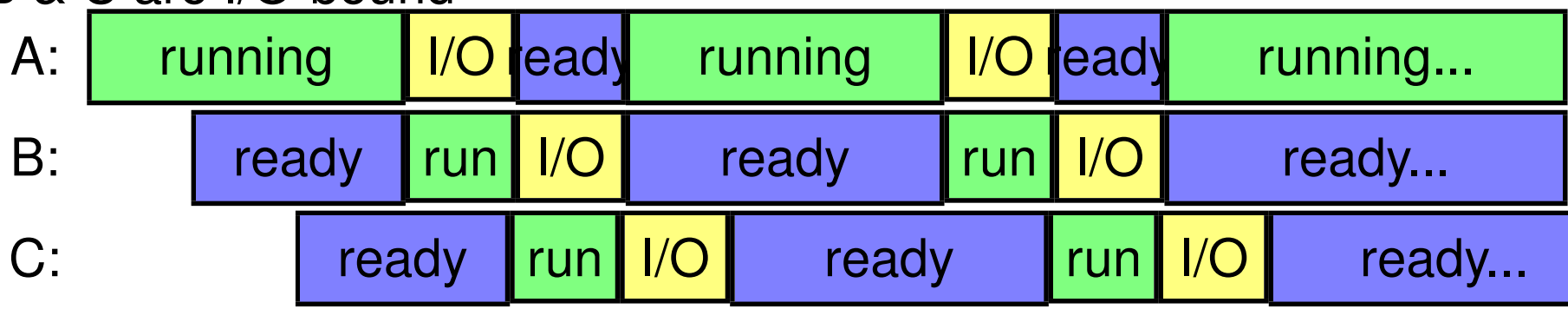
Average turnaround time: A,B (10+14)/2=12; B,A (14+4)/2=9;
Average waiting time: A,B (0+10)/2=5; B,A (4+0)/2=2;

First-Come-First-Served (FCFS)

Simplest CPU scheduling algorithm
First process in ready state gets CPU first & runs until blocked (or finished).

Requires a single queue of ready processes:
– add ready process to queue tail
– if the CPU is free, run process at queue head

e.g. A is CPU-bound
B & C are I/O-bound

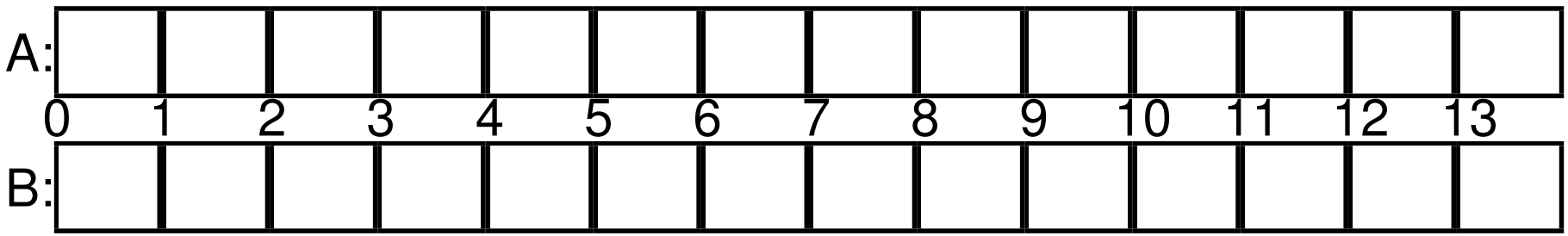


("I/O" = blocked, waiting for I/O to complete)
B and C spend too much time "ready"

Question - FCFS scheduling

Process A arrives at time 0: 4 time-units CPU, then 2 I/O, then 3 CPU, then 2 I/O

Process B arrives at time 1: 3 time-units CPU, then 1 I/O, then 1 CPU, then 1 I/O



Preemptive vs Non-Preemptive Scheduling

Non-preemptive scheduling: process runs until terminated or "blocked"

To avoid a process with a very long CPU-burst hogging CPU:

Preemptive scheduling: process can run (continuously) for some fixed maximum time;
if it reaches the maximum time, it is interrupted & set "ready"
and the scheduler runs another process

needs timer interrupt

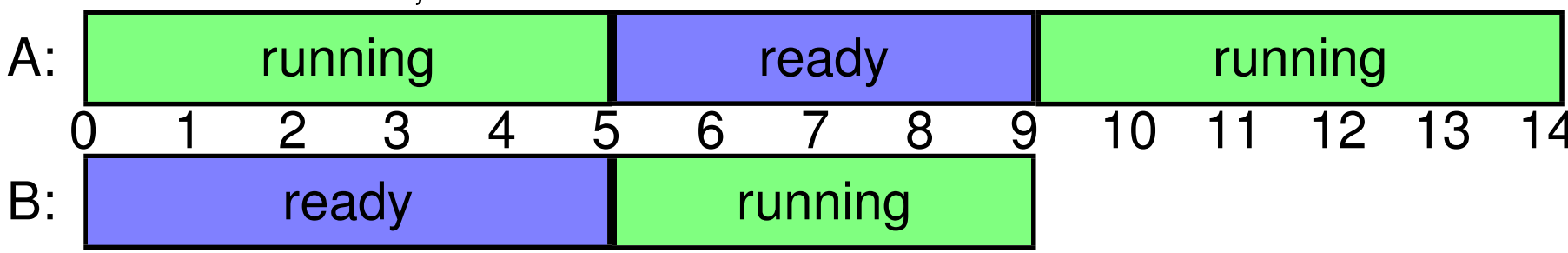
fixed amount of time = "**time-slice**" or "**time quantum**"
e.g. 10-100msec

Appropriate length for time quantum?

A preemptive version of the first example

Two processes, A and B, arrive at time 0;
CPU-burst time length: A=10, B=4 time-units
time-slice = 5 time-units

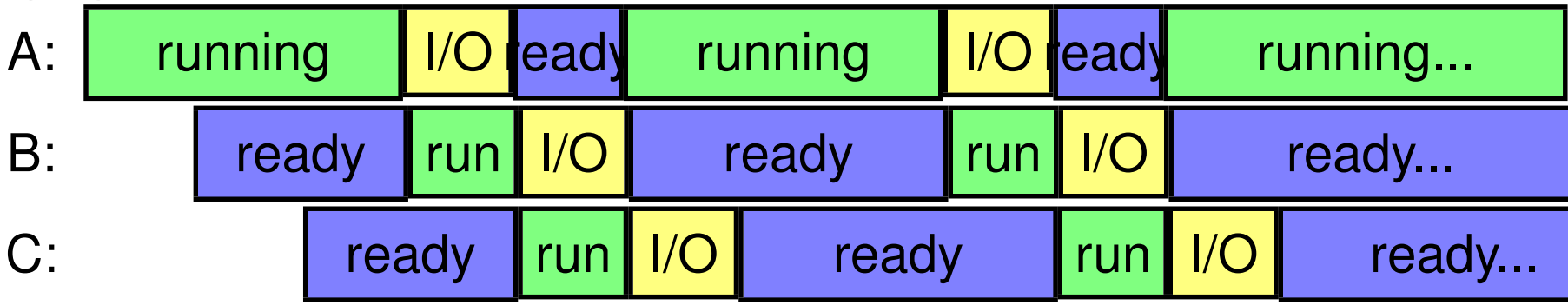
3rd case: A then B, & time-slice=5



Average turnaround time: 3rd: (14+9)/2=11.5 (was 12 or 9)
Average waiting time: 3rd: (4+5)/2=4.5 (was 5 or 2)

Round-Robin Scheduling

FCFS:



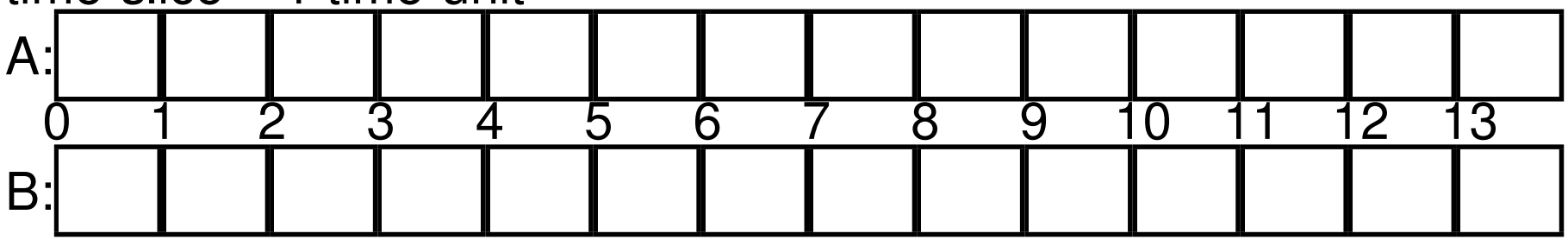
time-sliced; preempted process to ready-queue tail; run head

Simplest algorithm for time-sharing systems
Improves average turnaround time & waiting time

Question - Round-Robin scheduling

Process A arrives at time 0: 4 time-units CPU, then 2 I/O, then 3 CPU, then 2 I/O
Process B arrives at time 1: 3 time-units CPU, then 1 I/O, then 1 CPU, then 1 I/O

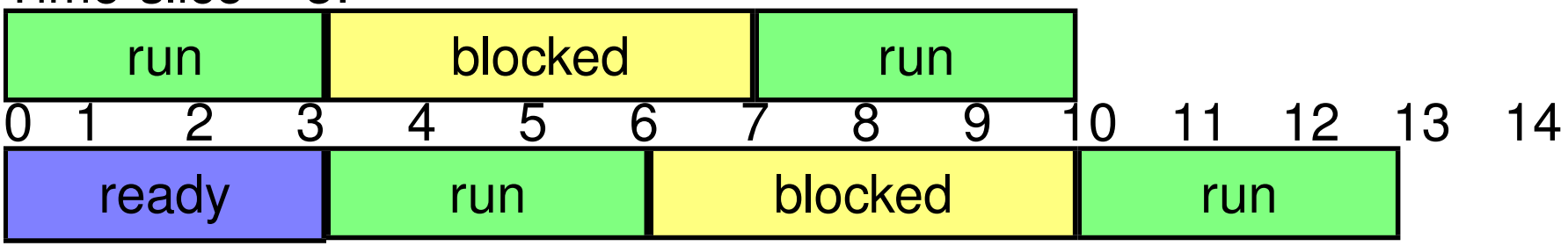
time-slice = 1 time-unit



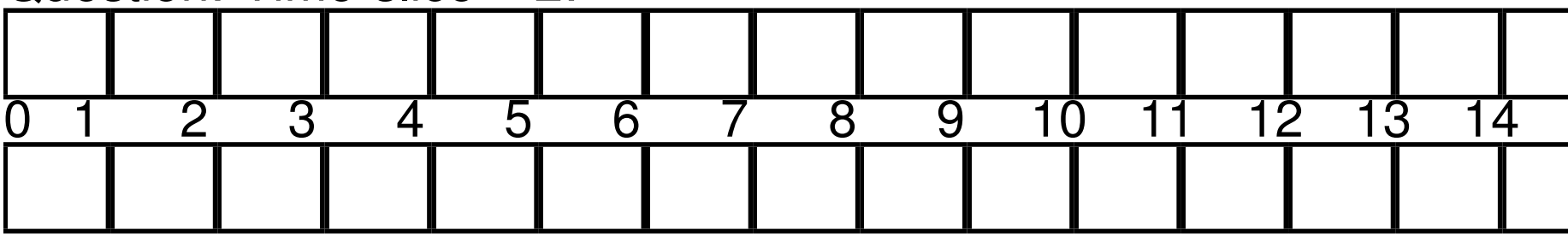
Length of time-slice?

Two identical processes: 3 time-units CPU, then 4 time-units I/O, then 3 time-units CPU

Time-slice = 3:



Question: Time-slice = 2:



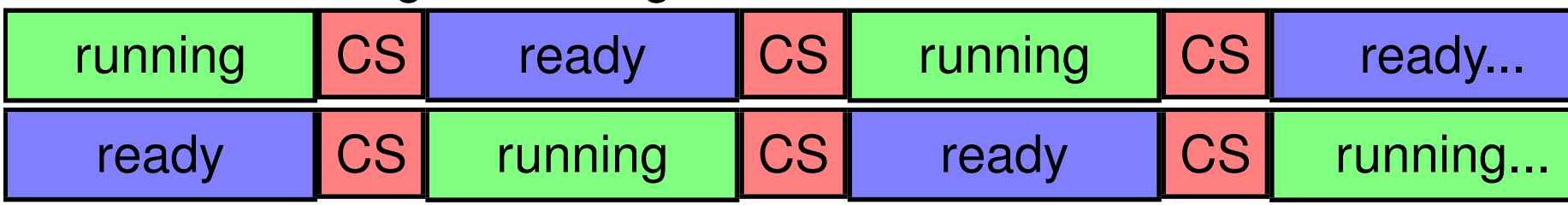
Scheduling Goals / Criteria for Evaluation

General, chose scheduling algorithms with most typical behaviour that best satisfies most desirable criteria:

– **Fairness**
– high **CPU utilisation**
– high **Throughput**
– low **Turnaround time**
– low **Waiting time**
– low **Response time**
– **Meeting deadlines**
– **Prioritisation**
... etc.

What is a good choice for the time quantum?

Different time slices may lead to different results (e.g. above)
The major trade-off is the cost of a context switch – ignored so far.
(i.e. save process state, pick new process, restore its state)
If the time slice is smaller, the cost of the context switch (CS) becomes more significant e.g.:



time-slice:context-switch = 3:1 $\rightarrow$ 25% CPU time lost

Two solutions:
- increase the time slice (but too much will make Round-Robin look like FCFS);
- reduce the cost of context switch (H/W support?)
e.g. quantum = 20-50ms, context switch $<$ 1ms

Non-preemptive – Shortest-Job-First

In the first example, starting with shortest job minimised average turnaround & waiting time.

Generalise: given a set of ready processes, run the one that has the smallest CPU burst

e.g. processes A, B, C, D arrive at time 0,
CPU bursts: A=7, B=4, C=9, D=5

Summary of key points

Scheduler: chooses a process from the ready state to run

Preemptive & Non-Preemptive

Algorithms: FCFS, Round-Robin, Shortest-First
– typical behaviour?

Criteria: fairness, utilisation, throughput, turnaround, wait, response, ...
– most desirable in given situation?

Next lecture: more process scheduling – (priority-based)

Your Questions

For next time

Explain why the time slice in pre-emptive process scheduling algorithms is normally significantly longer than the time needed for a context switch (2 marks)

Why is a schedule giving lowest average turnaround time the same as that giving lowest average waiting time? (1 mark)

Given a set of jobs with known processing time, all available to run, explain why repeatedly running the shortest job next gives the lowest average turnaround time. (3 marks)

What is a CPU burst and an I/O burst? What is a CPU-bound and an I/O bound process? Why is it a good strategy in process scheduling to give higher priority to I/O bound processes? (4 marks)

Exam Questions

Three identical processes arrive at time 0, 3, 6. Each requires 3 time-units of CPU, then 6 time-units of I/O, then 3 time-units of CPU. Using FCFS scheduling, what is the maximum amount of time that can be spent for each context switch so that all processes will have finished in at most 20 time units? (5 marks)

A process manager needs to run a total of 100 processes, each needing 3 seconds of CPU time.
i) Explain why no scheduling algorithm could complete all 100 processes in less than 5 minutes on 1 CPU. (2 marks)
ii) Using FCFS scheduling, calculate the average turnaround time and average waiting time of these processes. (3 marks)

Glossary

Scheduling
CPU burst
I/O burst
CPU bound
I/O bound
First-Come-First-Served (FCFS) Scheduling
(Average) Turnaround time
(Average) Waiting time
Non-preemptive scheduling
Preemptive scheduling
Time-slice, Time quantum
Round-Robin Scheduling
Shortest-Job-First Scheduling (SJF)
Scheduling Fairness
CPU utilisation
Throughput
Response time

Reading

OSC/J: sections 5.1, 5.2, 5.3.1, 5.3.4

older OSC/J: sections 6.1, 6.2, 6.3.1, 6.3.4

MOS2: sections 2.5, 2.5.1, first 2 pages of 2.5.2 and of 2.5.3 (round-robin)

MOS3: sections 2.4, 2.4.1, first 2 pages of 2.4.2 and of 2.4.3 (round-robin)