1.2.1 Scheduling

Processor Scheduling

In order for a computer to be multi-tasking and be able to run multiple applications simultaneously, the operating system needs to allocate the limited processor time to each one in a way that:

- Maximises the efficient use of the processor.
- Ensures processes receive a fair share of processor time and have reasonable response times.
- Keeps processor and other hardware resources as busy as possible
- Prevents process starvation, where a process waits indefinitely and never gets processor time

Scheduling algorithms

Round Robin

Processing jobs are placed in a queue, and each process is given a fixed time slice (quantum) of processor time.

If a process does not finish during its time slice, it is moved to the back of the queue and will receive another time slice later.

First come first served

Jobs are processed to completion in the order in which they arrive. No priority is given and short processes may have to wait behind longer processes, resulting in poor response times.

Multi-level feedback queues

Multiple queues are used to hold processes at different priority levels. Processes can move between queues depending on factors such as how much processor time they have used and how long they have been waiting.

Higher-priority queues are normally given processor time before lower-priority queues.

A process that uses a large amount of processor time may be moved to a lower-priority queue, while a process that has waited for a long time may be moved to a higher-priority queue.

Shortest job first

The process with the estimated shortest running time is selected next and runs until completion. Longer processes may experience starvation if short processes continue to arrive.

Shortest remaining time

Shortest remaining time is a pre-emptive scheduling algorithm. The process with the shortest estimated remaining time is selected. If a new process arrives with a shorter remaining time than the currently running process, the current process is paused and the new process is run.

(**Pre-emptive** means the operating system is allowed to interrupt a process that is currently running.)

Comparison Table

Algorithm	How it selects processes	Pre-emptive?	Main issue/benefit
Round Robin	Processes take turns using a fixed time slice	Yes	Fair; good response times
First Come First Served	First process to arrive runs first	No	Simple, but short processes can wait behind long ones
Multi-level Feedback Queue	Processes are placed in priority queues and can move between queues	Typically yes	Can prioritise processes and reduce starvation
Shortest Job First	Shortest estimated running time runs first	No	Efficient average waiting time, but long processes may starve
Shortest Remaining Time	Shortest estimated remaining time runs first	Yes	Can respond quickly to short processes, but longer processes may be repeatedly interrupted

(**Pre-emptive** means the operating system is allowed to interrupt a process that is currently running.)