Scheduling Concepts (Worked Example)

We'll use three processes:

- **A:** Burst time = 6
- **B:** Burst time = 4
- **C:** Burst time = 8
- All arrive at time 0.

FCFS (First-Come, First-Served)

- Order: A → B → C
- Completion times:
  - A = 6
  - B = 10
  - C = 18
- Turnaround times (TAT = completion − arrival):
  - A = 6
  - B = 10
  - C = 18
- Waiting times (WT = TAT − burst):
  - A = 0
  - B = 6
  - C = 10
- Averages:
  - Avg TAT = (6 + 10 + 18) ÷ 3 = 11.33
  - Avg WT = (0 + 6 + 10) ÷ 3 = 5.33

Round Robin (Quantum = 4)

- Order slices:
  - A: 4 (remaining 2), B: 4 (done), C: 4 (remaining 4), A: 2 (done), C: 4 (done).
- Completion times:
  - B = 8
  - A = 10
  - C = 18
- Turnaround times:
  - A = 10
  - B = 8

    - C = 18
- Waiting times:
    - A = 10 − 6 = 4
    - B = 8 − 4 = 4
    - C = 18 − 8 = 10
- Averages:
    - Avg TAT = (10 + 8 + 18) ÷ 3 = 12
    - Avg WT = (4 + 4 + 10) ÷ 3 = 6

Reflection (Week 5)

This week's exercises helped me understand how operating systems manage processes and decide which tasks run at a given time. By experimenting with foreground and background jobs, I saw how the shell can control multiple processes and how the OS tracks them. Adjusting priorities with `nice` and `renice` showed me how scheduling can be influenced manually, and how CPU usage changes when a process has lower priority.

Working through FCFS and Round Robin scheduling examples gave me a clearer picture of how waiting time and turnaround time are calculated. I realized that FCFS is simple but can cause long waiting times for later processes, while Round Robin improves fairness by giving each process a time slice.

Overall, I learned that process scheduling is about balancing efficiency and fairness. These concepts are not just theoretical — they directly affect system performance and responsiveness. I feel more confident now in analyzing scheduling problems and using Linux commands to observe and control processes.