Operating Systems (part 2) Exercises

These exercises are grouped into three parts. Part I contains routine exercises to help you understand the ideas directly presented in lectures. The exercises in Part II are designed to grow and deepen your understanding of principles that underpin Operating Systems. These exercises extend the lecture material and invite you to think about 'why' questions and about optimisations to what was lectured as well as alternatives to what was lectured. Part III are open-ended, stretching questions that you could tackle in your mini research project.

I hope you enjoy working through these questions, puzzles and research questions!

John Fawcett, July 2023

Part I

1. Why do we need to virtualise the printer? Why not allow processes to send their documents straight to the real printer?
2. Shortest Job First (SJF), Shortest Remaining Time First (SRTF) and Round Robin (RR) are CPU scheduling algorithms. Briefly describe each.

   Consider the following set of processes.

Process	Creation Time	Required Computing Time
P_1	0	25
P_2	5	15
P_3	10	5
P_4	15	5

   a. Draw a diagram showing how SJF would schedule the processes. What is the average waiting time?
   b. Draw a diagram showing how SRTF would schedule the processes. What is the average waiting time?
   c. Assuming a quantum of 10 time units, draw a diagram showing how RR would schedule the processes. What is the average waiting time in this case?
   d. It is suggested that RR could be improved by reducing the quantum to 1 time unit. What are the advantages and disadvantages of this proposal?

3. With the Unix process model (as presented on slide 5), how could the operating system take advantage of a multicore CPU?
4. If you can run Linux or MacOS on your computer and have some experience with the C programming language, write a C program that calls fork(), then has the two processes sleep for 2s and 3s respectively before printing "Hello 1" in one of them and "Hello 2" in the other.
5. Write a program that adds 1 to a variable N times. Time the program for different values of N: 1, 1000, 5,000, 1,000,000, 5,000,000, 10,000,000, 50,000,000, 100,000,000, 500,000,000, 1,000,000,000. Plot a graph and explain why the points do not lie on a perfectly straight line.
6. If you can run Linux or MacOS on your computer, have a look in /proc/interrupts, research what the values mean, and explain what the information tells you about what your computer has done since it booted up.

Part II

1. Why might it be difficult for a process to claim its fair share of CPU time if it also uses its fair share of hard disk time?
2. Why do we need an EXIT state in the Unix process model? Why not just get rid of a process that has finished?

Part III

1. We talked about keeping commands for the hard disk in a *queue* data structure, which implements first-come-first-served (FCFS). Each of the following would be an interesting research project:
    a. Compare FCFS to the SCAN algorithm.
    b. Compare FCFS to the C-SCAN algorithm.
    c. Compare FCFS to the LOOK algorithm.
    d. Compare FCFS to the CLOOK algorithm.
    e. Compare FCFS to the FSCAN algorithm.
2. Research how the Windows NT (4.0) CPU scheduling algorithm works.
3. How does the Unix dynamic priority scheduler work?