Operating Systems - Review Questions 1

Deadline: Nov. 18, 2023

Students:
Hiba Qutbuddin Habib
Ayah Babiker

1. **When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process? Stack, Heap, or Shared memory segments?**

The shared Memory is shared between the parent and child process.

2. Explain what the output will be at LINE A.

```
int value = 15;
int main() {
  pid t pid;
  pid = fork();

  if (pid == 0) { /* child process */
    value += 10;
    return 0;
  } else if (pid > 0) { /* parent process */
    wait(NULL);
    printf("PARENT: value = %d",value); /* LINE A */
    return 0;
  }
}
```

The LINE A has the parent value, which will be 15. Because we have done nothing to that value as the parent process will just wait until the child finishes his work and then get the original value which is the initiate value15.

3. **How many processes are created in the following code?**

```
int main() {
  fork();
  fork();
  fork();
}
```

The total number of processes is 2^n, and n is the number of the fork system calls.
In this case the total number of processes will be 2^3= 8.

4. **See Section 4.1 of the “Operating Systems Concepts” book. Does the multithreaded web server described in that section exhibit task or data parallelism?**

Task parallelism means that each process is executing different tasks. Whereas data parallelism means that the same task is being executed but the data that it is using to execute is different in each process. In section 4.1 you can see an example of data parallelism.

5. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

One difference is that user-level threads are initiated by the user whereas kernel-level threads are handled by the operating system thereby the names for the threads. The other difference is related to block operation, iif one user level thread performs blocking operation will result in that the entire process will be blocked, but when it comes to the kernel thread so another thread can continue the execution when one kernel thread performs a blocking operation. In general user-level threads are much easier to manage in comparison to kernel-level threads.

It would be more advantageous to use user-level threads when you need to be able to communicate with any type of operating system since the kernel-level threads are operating system specific.

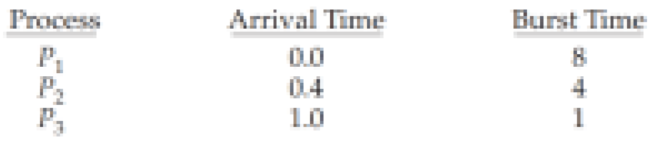
6. Describe the actions taken by a kernel to context-switch between kernel level threads.

During a context switch there is usage of a PCB to save the states of the processes. So the first process state will be saved so that you then can continue to another process and continue on that one and then save its progress in a pcb. This means that you can go from one process to another by saving the progress so that you later can go back to it to continue.

7**. Explain the difference between preemptive and nonpreemptive scheduling.**

For Non-preemptive scheduling , the CPU is allocated to a process until it completes its work or enters a waiting state, whereas Preemptive scheduling allocates the CPU to processes for a limited time and can interrupt an ongoing process if necessary. Preemptive scheduling provides flexibility by allowing interruptions, while Non-preemptive scheduling is more rigid and does not allow interruptions until the process completes or reaches its time limit. This can affect response times, with Preemptive scheduling typically resulting in shorter response times, while Non-preemptive scheduling may lead to longer response times.

8. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.



Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

Using the FCFS scheduling algorithm:

- P1 finishes at 8 (8+0=8) and has turnaround time of 8 - 0 = 8
- P2 cannot start until P1 is done and that why it finishes at 12 (8+4=12) and it has a turnaround time of 12 - 0.4 = 11.6
- P3 starts also when P2 is done, so it finishes at 13 (12+1=13) and it has turnaround time of 13 - 1 = 12
- **The average turnaround time therefore is :**
  Average turnaround = $(8 + 11.6 + 12) \div 3 = 10.53$

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

Using SJF scheduling algorithm:

- P1 is the only available at time 0,0 and that's why it runs first. The finishes at 8 (8+0=8) and has turnaround time of 8 - 0 = 8
- When P1 is done, P2 and P3 arrive. But because P3 has the shortest burst time so it runs first. P3 finishes at 9 (8+1=9) and it has a turnaround time of 9 - 1 = 8
- P2 will run finally because its the only process left, it finishes at 13 (9+4=13) and it has turnaround
  time of 13 - 0.4 = 12,6
- **The average turnaround time therefore is :**
  Average turnaround = $(8 + 8 + 12.6) \div 3 = 9.53$

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling

Since the CPU is idle until time 1.0 and at time 1.0 all the three process has arrived:

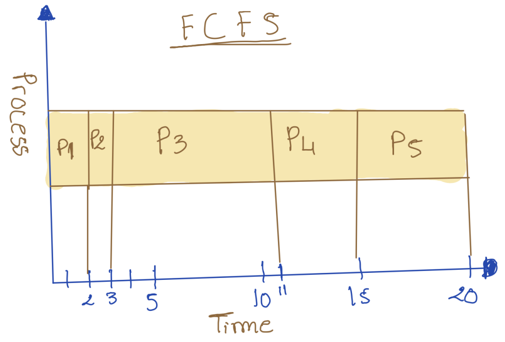
- P3 has the shortest burst time so it starts running at time 1.0. The finishes at 2(1+1=2) and has turnaround time of 2 - 1 = 1
- P2 has the next shortest burst time so it starts running at 2 and finishes at 6 (2+4=6) and it has a turnaround time of 6 - 0.4 = 5.6
- P1 runs since its the only process left and finishes at 14(6+8=14) and it has turnaround time of 14 - 0.0 = 14
- **The average turnaround time therefore is :**
  Average turnaround = $(1 + 5.6 + 14) \div 3 = 6.87$

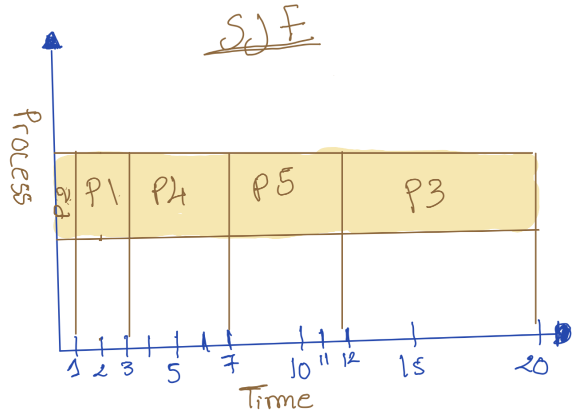
9. Consider the following set of processes, with the length of the CPU burst time given in milliseconds

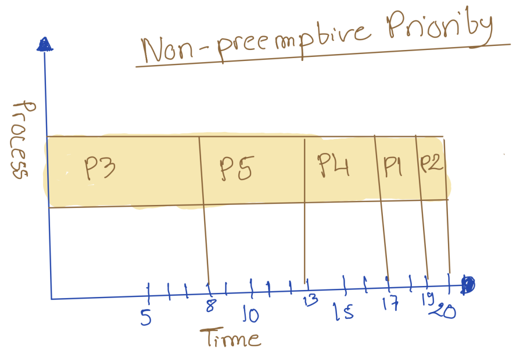
Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

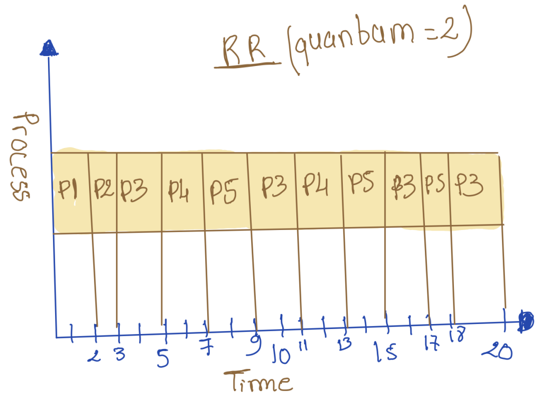
The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).









b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these scheduling

algorithms?

- FCFS:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	2	2 + 0 = 2	2 - 0 = 2	2 - 2 = 0
P2	0	1	2 + 1 = 3	3 - 0 = 3	3 - 1 = 2
P3	0	8	3 + 8 = 11	11 - 0 = 11	11 - 8 = 3
P4	0	4	11 + 4 = 15	15 - 0 = 15	15 - 4 = 11
P5	0	5	15 + 5 = 20	20 - 0 = 20	20 - 5 = 15

- SJF:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P2	0	1	1 + 0 = 1	1 - 0 = 1	1 - 1 = 0
P1	0	2	1 + 2 = 3	3 - 0 = 3	3 - 2 = 1
P4	0	4	3 + 4 = 7	7 - 0 = 7	7 - 4 = 3
P5	0	5	7 + 5 = 12	12 - 0 = 12	12 - 5 = 7
P3	0	8	12 + 8 = 20	20 - 0 = 20	20 - 8 = 12

- Non preemptive priority:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P3	0	8	8 + 0 = 8	8 - 0 = 8	8 - 8 = 0
P5	0	5	8 + 5 = 13	13 - 0 = 13	13 - 5 = 8
P4	0	4	13 + 4 = 17	17 - 0 = 17	17 - 4 = 13
P1	0	2	17 + 2 = 19	19 - 0 = 19	19 - 2 = 17
P2	0	1	19 + 1 = 20	20 - 0 = 20	20 - 1 = 19

- RR(Quantum = 2):

Process	Arrival Time	Burst Time	Turnaround Time	Waiting Time
P1	0	2	2	0
P2	0	1	3	2
P3	0	8	20	3 + (9 - 5) + (15 - 11) + (18 - 17) = 12
P4	0	4	13	5 + (11 - 7) = 9
P5	0	5	18	7 + (13 - 9) + (17 - 15) = 13

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Algorithm	Average Waiting Time
FCFS	(0 + 2 + 3 + 11 + 15) / 5 = 6.2
SJF	(0 + 1 + 3 + 7 + 12) / 5 = 4.6
Non Preemptive Priority	(0 + 8 + 13 + 17 + 19) / 5 = 11.4
RR	(0 + 2 + 12 + 9 + 13) / 5 = 7.2

The algorithm that has the minimum average waiting time is SJF(4.6)