Bilkent University
Fall 2023-2024

Department of Computer Engineering

CS342
Operating Systems

Homework #2

Deniz Tuna Onguner
22001788

Section 1
İbrahim Körpeoğlu

27 October 2023

Question #1

(a) There are a total of **three** (3) processes, *including the parent/main process*, are expected to be created and then uncreated. Initially, the parent process is created and it creates a child process with the very first `fork()` and later this child process created creates another child process with the second `fork()`.

(b) The values of 10, 210, 215, 110, 115, 160 (without being in any particular order) are going to be printed out onto the console.

(c) The very first value printed will always be 10 regardlessly. And, the rest of the values (provided in part b) will satisfy the following assertions, respectively:

1. 110 will always be printed before 160.
2. 110 will also be printed before 115 all the time.
3. 210 will always be printed before 215.

So, there are 20 permutations satisfying all these conditions, and they are as the follows — *10 is excluded since it always will be the very first one appears*.

1. (210, 215, 110, 115, 160)
2. (210, 215, 110, 160, 115)
3. (210, 110, 215, 115, 160)
4. (210, 110, 215, 160, 115)
5. (210, 110, 115, 215, 160)
6. (210, 110, 115, 160, 215)
7. (210, 110, 160, 215, 115)
8. (210, 110, 160, 115, 215)
9. (110, 210, 215, 115, 160)
10. (110, 210, 215, 160, 115)
11. (110, 210, 115, 215, 160)
12. (110, 210, 115, 160, 215)
13. (110, 210, 160, 215, 115)
14. (110, 210, 160, 115, 215)
15. (110, 115, 210, 215, 160)
16. (110, 115, 210, 160, 215)
17. (110, 115, 160, 210, 215)
18. (110, 160, 210, 215, 115)
19. (110, 160, 210, 115, 215)
20. (110, 160, 115, 210, 215)

Please note that those above are all the possible orders but the frequency of them appearing on an actual execution may or may not be equal to one other.

Question #2

(a) The output that will be generated (the values that will be printed out) is as the follows 1000, 2000, 1100, 400, 1100, 2000 being in the exact order that will appear on an actual runtime.
(b) There is only one possible order (refer to part a, question #2) because the **pthread_join** function in the main thread is blocking. The main thread will wait for the foo thread to complete before printing the final values. Therefore, the execution order is deterministic.
(c) Moving the **pthread_join** call at the very end of the main function could result in a different order of the final output; however, it is certainly not expected that there will be any other additional values printed different from the former executions of the code.

Question #3

For this particular question/situation, we can apply to *Amdahl's Law*. Amdahl's Law is a formula that identifies potential performance gains from adding additional computing cores to an application that has both serial (nonparallel) and parallel components. If *S* is the portion of the application that must be performed serially on a system with *N* processing cores, the formula appears as follows:

$$speedup \leq \frac{1}{S + \frac{(1 - S)}{N}}$$ [1]

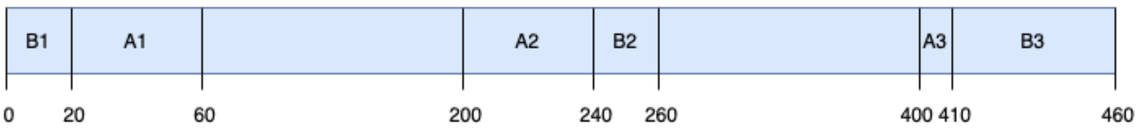
So, when we write the given parameters on the equation above, we observe the followings:

$$\frac{1}{25\% + \frac{(1 - 25\%)}{4}} \approx 2.28$$ where the computer has 4 cores of CPU.

$$\frac{1}{25\% + \frac{(1 - 25\%)}{8}} \approx 2.90$$ where the computer has 8 cores of CPU.

Question #4

(a)



(b) The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts. We can define the exponential average with the following formula:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

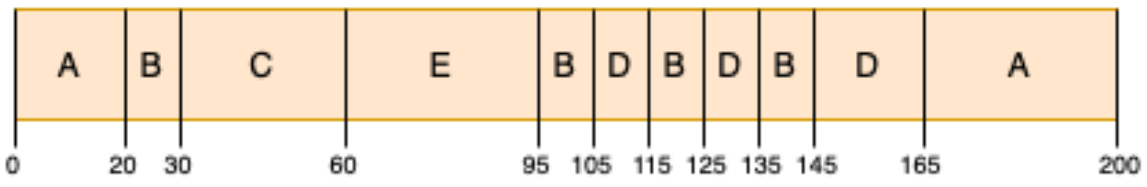
where t_n is the length of the nth CPU burst, and τ_{n+1} is the predicted value for the next CPU burst [1].

From this equation/formula, we can calculate the predicted values of A4 and B4, which are **17.5 ms** and **40 ms**, respectively.

Question #5

	Arv Time	Finish time	Turnaround time	Waiting time	Response time
A	0	200	200	155	0
B	15	145	130	90	5
C	25	60	35	5	5
D	30	165	135	95	75
E	35	95	60	25	25

The Gantt chart:



Question #6

(a) B will get 50% of the total CPU time, $\frac{300}{600}(100) = 50\,\%$.

(b) In this particular case, B is going to get $\frac{300}{800}(100) = 37.5\ \%$ of CPU time.

Question #7

(a) The worst-case utilization is calculated via the following equation:

$$N(2^{\frac{1}{N}} - 1)$$

Solving for $N = 4$, we get $4(2^{\frac{1}{4}} - 1) \approx 0.756 \approx 76\,\%$.

(b) Yes, it is very possible since we choose our utilization less than the worst-case.

70% < 76% (refer to part a, question #7).

Question #8

(a) $\frac{1}{\lambda} = (25)10^{-3}$, $\lambda = 40$ arrivals/sec; CPU utilization = $\frac{\lambda}{\mu} = \frac{40}{42} = 95\frac{5}{21}\,\%$.

(b) Average response time = $\frac{1}{\mu - \lambda} = \frac{1}{42 - 40} = \frac{1}{2}$ sec.

(c) $E[N] = (\lambda)\frac{1}{\mu - \lambda} - 1 = (40)\frac{1}{42 - 40} - 1 = 19.$

(d) $E[W] = E[R] - \frac{1}{\mu} \approx 0.48$ sec, where $E[R] = \frac{E[N]}{\lambda}$.

(e) $1 - p = 1 - \frac{20}{21} = \frac{1}{21} \approx 0.0476.$

(f) $\frac{1}{\mu - \lambda^{'}} = \frac{1}{2(\mu - \lambda)} \Rightarrow 2 = \frac{\mu - \lambda^{'}}{\mu - \lambda} = \frac{42 - \lambda^{'}}{42 - 40} \Rightarrow \lambda^{'} = 38$ arrivals/sec.

References

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, in *Operating system concepts*, 10th ed, Hoboken, NJ: Wiley, 2018, p. 164-208.