CS 314 Operating Systems, Spring 2022
Quiz #4 on February 22, 2022

List of the possible questions

#1: What do “sufficient conditions” guarantee?

#2: What do “necessary conditions” guarantee?

#3: If a sufficient condition is not satisfied, what conclusion can we draw?

#4: If a necessary condition is satisfied, what conclusion can we draw?

#5: [this question has been dropped] Describe how “micro-kernel architecture” and “non micro-kernel architecture” are different in how system calls issued by user applications will be executed.

#6: [this question has been dropped] What are the advantages in using “micro-kernel architecture”? What is the primary disadvantage in “micro-kernel architecture”?

#7: [this question has been dropped] What is “critical section”?

#8: [this question has been dropped] What is “mutual exclusion”?

#9: What does “atomic” in “atomic operations” mean?

#10: What are “counting semaphores” (how are “counting semaphores” different from “binary (“mutex”) semaphores”)?

#11: Who manages semaphores?

#12: Operating systems use “queue (FIFO data structure)” for managing processes blocked (waiting) on a semaphore. Why is FIFO-queue used (the best reason for using FIFO structure)?

#13: What is “process deadlock”?

#14: How is “process deadlock” different from “(process) starvation”?

#15: What are the two different types of “process deadlock”?

#16: [this question has been dropped] Why is “process deadlocks due to message passing” easier to solve than “process deadlocks due to resource conflict”?

#17: What are the four necessary conditions for a process deadlock to occur?
#18: Why is it difficult to eliminate the condition of “mutual exclusion” to prevent a process deadlock from occurring?

#19: Why is it difficult to eliminate the condition of “non preemptive resources” to prevent a process deadlock from occurring?

#20: [this question has been dropped] Why is it difficult to eliminate the condition of “hold & wait” (by applying “request all after you drop what all what you currently hold” method) to prevent a process deadlock from occurring?

#21: Which of the following sentence is the correct definition of “safe state” in deadlock avoidance (select the best option)?

(a) From the current state, deadlock can never happen no matter what happen after the current state.
(b) From the current state, there is at least one particular way that lets all the processes complete without causing a deadlock.
(c) The deadlock has already happened.
(d) If deadlock has not happened yet that is a safe state.
(e) None of the above

#22: [this question has been dropped] Why is it difficult to eliminate the condition of “circular wait” (by applying “assign resource IDS and require all processes to make requests in the ascending or the descending order” method) to prevent a process deadlock from occurring?

#23: What are the two states in “deadlock avoidance”?

#24: In deadlock prevention, one of the solutions is not to allow any process to “hold & wait” (if a process that holds some non-preemptive resources needs additional resource(s), the process must release all the resources it currently holds and then it requests all the resources (both what it has released and what it additionally needs). Explain how this will prevent deadlocks.

#25: In deadlock prevention, one of the techniques is not to allow processes to cause “circular wait” (processes must request resources in the ascending (or the descending) order of their resource IDs). What is the potential danger this approach entails?

#26: In the classroom, we discussed what we can do to make sure one of the four necessary conditions for a deadlock is not satisfied. Is it possible to have a technique that never causes “circular wait”? If yes, describe how.
#27: For the following situations, which one is (are) in deadlock?

In the above figures, Ri = Resource, Pi = Process, a directional edge from R to P is assignment and a directional edge from P to R is a blocked request (one edge for one resource). You don’t need to describe anything. If none of the three is in deadlock, clearly mention so. If the same resource is assigned to more than one process, it means that more than one instance of the resource are available.