CS 314 Operating Systems
Spring 2024
Quiz #6 on February 15, 2024 (SOLUTIONS)

Your Last Three Digits: _____
(please do NOT write all of your student ID or your name)

Grade: _____

****

(1) What is "process deadlock"? How is it different from "process starvation"?

Process deadlock is a situation where all the processes (those that look for a common resource) blocks each other (i.e. "wait for each other) in a way none of them can make any progress.

In "process starvation", there is at least one process that is making a progress. It means that it is still (logically) possible for all processes to eventually finish, while it is impossible once a process deadlock occurs.

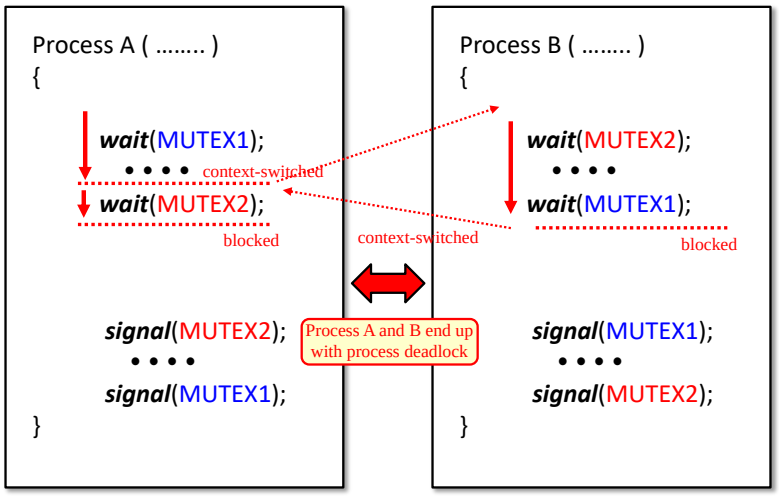
(2) Why is it difficult to eliminate the condition of "mutual exclusion" to prevent a process deadlock from occurring?

It is because some resources are inherently a mutually exclusive device (e.g., printers, DVD-burners, and etc.). While some resources are inherently mutually-exclusive. Converting them to non-mutually excusive devices is not a practical option.

(3) If more than one mutex (binary) semaphore is used, can process deadlock occur? If no, explain why not. If yes, explain how using an example.

YES. If two (or more) processes call/perform "wait" on two (or multiple) semaphores in a different order, processes can cause a process deadlock.

For example, let us assume that process A and B are going to call "wait" to two semaphores, MUTEX1 and MUTEX2. Right after A calls "wait" to MUTEX1 (assume that A "locks" MUTEX1), a context switching occurs, preempting a processor from A to B. Then, B calls "wait" to MUTEX2 (assume that B "locks" MUTEX2). If B calls "wait" to MUTEX1, B blocks on MUTEX1 (since A already locks on MUTEX1). Then, a processor is context-switched from B to A, allowing A to call "wait" to MUTEX2. This way, A and B caused a process deadlock.



(4) Why is it difficult to eliminate the condition of "non preemptive resources" to prevent a process deadlock from occurring?

It is because some resources are inherently a non-preemptive device (e.g., DVD-burners). While some resources are inherently non-preemptive, converting them to a preemptive devices is not a practical option.

(5) 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?

The "all or nothing approach" we discussed in our classroom will cause starvation especially to those processes that require a large number of non-preemptive resources that require mutually exclusive uses.