

# Process Synchronization



## Practice Exercises

- 6.1 In Section 6.4, we mentioned that disabling interrupts frequently can affect the system's clock. Explain why this can occur and how such effects can be minimized.

**Answer:** The system clock is updated at every clock interrupt. If interrupts were disabled—particularly for a long period of time—it is possible the system clock could easily lose the correct time. The system clock is also used for scheduling purposes. For example, the time quantum for a process is expressed as a number of clock ticks. At every clock interrupt, the scheduler determines if the time quantum for the currently running process has expired. If clock interrupts were disabled, the scheduler could not accurately assign time quanta. This effect can be minimized by disabling clock interrupts for only very short periods.

- 6.2 **The Cigarette-Smokers Problem.** Consider a system with three *smoker* processes and one *agent* process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper, and matches. One of the smoker processes has paper, another has tobacco, and the third has matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats. Write a program to synchronize the agent and the smokers using Java synchronization.

**Answer:** Please refer to the supporting Web site for source code solution.

- 6.3 Explain why Solaris, Windows XP, and Linux implement multiple locking mechanisms. Describe the circumstances under which they use spinlocks, mutexes, semaphores, adaptive mutexes, and condition variables. In each case, explain why the mechanism is needed.

**Answer:** These operating systems provide different locking mechanisms depending on the application developers' needs. Spinlocks are

useful for multiprocessor systems where a thread can run in a busy-loop (for a short period of time) rather than incurring the overhead of being put in a sleep queue. Mutexes are useful for locking resources. Solaris 2 uses adaptive mutexes, meaning that the mutex is implemented with a spin lock on multiprocessor machines. Semaphores and condition variables are more appropriate tools for synchronization when a resource must be held for a long period of time, since spinning is inefficient for a long duration.

- 6.4 List three examples of deadlocks that are not related to a computer-system environment.

**Answer:**

- Two cars crossing a single-lane bridge from opposite directions.
- A person going down a ladder while another person is climbing up the ladder.
- Two trains traveling toward each other on the same track.

- 6.5 Is it possible to have a deadlock involving only one single process? Explain your answer.

**Answer:** No. This follows directly from the hold-and-wait condition.