CHAPTER 6

Process Synchronization

Practice Exercises

6.1 In Section 6.4 we mentioned that disabling interrupts frequently could affect the system's clock. Explain why it could and how such effects could 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 quantums. 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 Give the reasons 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 Explain the differences, in terms of cost, among the three storage types volatile, nonvolatile, and stable.

Answer: Volatile storage refers to main and cache memory and is very fast. However, volatile storage cannot survive system crashes or powering down the system. Nonvolatile storage survives system crashes and powered-down systems. Disks and tapes are examples of nonvolatile storage. Recently, USB devices using erasable program read-only memory (EPROM) have appeared providing nonvolatile storage. Stable storage refers to storage that technically can *never* be lost as there are redundant backup copies of the data (usually on disk).

6.5 Explain the purpose of the checkpoint mechanism. How often should checkpoints be performed? Describe how the frequency of checkpoints affects:

- System performance when no failure occurs
- The time it takes to recover from a system crash
- The time it takes to recover from a disk crash

Answer: A checkpoint log record indicates that a log record and its modified data has been written to stable storage and that the transaction need not to be redone in case of a system crash. Obviously, the more often checkpoints are performed, the less likely it is that redundant updates will have to be performed during the recovery process.

- System performance when no failure occurs—If no failures occur, the system must incur the cost of performing checkpoints that are essentially unnecessary. In this situation, performing checkpoints less often will lead to better system performance.
- The time it takes to recover from a system crash—The existence of a checkpoint record means that an operation will not have to be redone during system recovery. In this situation, the more often checkpoints were performed, the faster the recovery time is from a system crash.
- The time it takes to recover from a disk crash—The existence of a checkpoint record means that an operation will not have to be redone during system recovery. In this situation, the more often

checkpoints were performed, the faster the recovery time is from a disk crash.

6.6 Explain the concept of transaction atomicity.
Answer: A transaction is a series of read and write operations upon some data followed by a commit operation. If the series of operations in a transaction cannot be completed, the transaction must be aborted and the operations that did take place must be rolled back. It is important that the series of operations in a transaction appear as one indivisible operation to ensure the integrity of the data being updated. Otherwise, data could be compromised if operations from two (or more) different transactions were intermixed.

6.7 Show that some schedules are possible under the two-phase locking protocol but not possible under the timestamp protocol, and vice versa.
Answer: A schedule that is allowed in the two-phase locking protocol but not in the timestamp protocol is:

step	T_0	T_1	Precedence
1	**lock-S**(A)		
2	**read**(A)		
3		**lock-X**(B)	
4		**write**(B)	
5		**unlock**(B)	
6	**lock-S**(B)		
7	**read**(B)		$T_1 \rightarrow T_0$
8	**unlock**(A)		
9	**unlock**(B)		

This schedule is not allowed in the timestamp protocol because at step 7, the W-timestamp of B is 1.

A schedule that is allowed in the timestamp protocol but not in the two-phase locking protocol is:

step	T_0	T_1	T_2
1	**write**(A)		
2		**write**(A)	
3			**write**(A)
4	**write**(B)		
5		**write**(B)	

This schedule cannot have lock instructions added to make it legal under two-phase locking protocol because T_1 must unlock (A) between steps 2 and 3, and must lock (B) between steps 4 and 5.

6.8 The `wait()` statement in all Java program examples was part of a `while` loop. Explain why you would always need to use a `while` statement when using `wait()` and why you would never use an `if` statement.
Answer: This is an important issue to emphasize! Java only provides anonymous notification—you cannot notify a certain thread that a cer-

tain condition is true. When a thread is notified, it is its responsibility to re-check the condition that it is waiting for. If a thread did not re-check the condition, it might have received the notification without the condition having been met.