CHAPTER 8

Deadlocks

Practice Exercises

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

8.2 Suppose that a system is in an unsafe state. Show that it is possible for the threads to complete their execution without entering a deadlocked state.

8.3 Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix ***Need***?

b. Is the system in a safe state?

c. If a request from thread T_1 arrives for (0,4,2,0), can the request be granted immediately?

8.4 A possible method for preventing deadlocks is to have a single, higher-order resource that must be requested before any other resource. For example, if multiple threads attempt to access the synchronization objects $A \cdots E$, deadlock is possible. (Such synchronization objects may include mutexes, semaphores, condition variables, and the like.) We can prevent deadlock by adding a sixth object F. Whenever a thread wants to acquire the synchronization lock for any object $A \cdots E$, it must first acquire the lock for object F. This solution is known as **containment**: the locks for objects $A \cdots E$ are contained within the lock for object F. Compare this scheme with the circular-wait scheme of Section 8.5.4.

8.5 Prove that the safety algorithm presented in Section 8.6.3 requires an order of $m \times n^2$ operations.

8.6 Consider a computer system that runs 5,000 jobs per month and has no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about ten jobs per deadlock. Each job is worth about two dollars (in CPU time), and the jobs terminated tend to be about half done when they are aborted.

A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in the system with an increase of about 10 percent in the average execution time per job. Since the machine currently has 30 percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average.

a. What are the arguments for installing the deadlock-avoidance algorithm?

b. What are the arguments against installing the deadlock-avoidance algorithm?

8.7 Can a system detect that some of its threads are starving? If you answer "yes," explain how it can. If you answer "no," explain how the system can deal with the starvation problem.

8.8 Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any threads that are blocked waiting for resources. If a blocked thread has the desired resources, then these resources are taken away from it and are given to the requesting thread. The vector of resources for which the blocked thread is waiting is increased to include the resources that were taken away.

For example, a system has three resource types, and the vector ***Available*** is initialized to (4,2,2). If thread T_0 asks for (2,2,1), it gets them. If T_1 asks for (1,0,1), it gets them. Then, if T_0 asks for (0,0,1), it is blocked (resource not available). If T_2 now asks for (2,0,0), it gets the available one (1,0,0), as well as one that was allocated to T_0 (since T_0 is blocked). T_0's ***Allocation*** vector goes down to (1,2,1), and its ***Need*** vector goes up to (1,0,1).

a. Can deadlock occur? If you answer "yes," give an example. If you answer "no," specify which necessary condition cannot occur.

b. Can indefinite blocking occur? Explain your answer.

8.9 Consider the following snapshot of a system:

	Allocation	*Max*
	A B C D	*A B C D*
T_0	3 0 1 4	5 1 1 7
T_1	2 2 1 0	3 2 1 1
T_2	3 1 2 1	3 3 2 1
T_3	0 5 1 0	4 6 1 2
T_4	4 2 1 2	6 3 2 5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. ***Available*** = (0, 3, 0, 1)

b. ***Available*** = (1, 0, 0, 2)

8.10 Suppose that you have coded the deadlock-avoidance safety algorithm that determines if a system is in a safe state or not, and now have been asked to implement the deadlock-detection algorithm. Can you do so by simply using the safety algorithm code and redefining $\boldsymbol{Max_i} = \boldsymbol{Waiting_i} + \boldsymbol{Allocation_i}$, where $\boldsymbol{Waiting_i}$ is a vector specifying the resources for which thread *i* is waiting and $\boldsymbol{Allocation_i}$ is as defined in Section 8.6? Explain your answer.

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