Deadlocks

Exercises

8.12 Consider the traffic deadlock depicted in Figure 8.12.

a. Show that the four necessary conditions for deadlock hold in this example.

b. State a simple rule for avoiding deadlocks in this system.

8.13 Draw the resource-allocation graph that illustrates deadlock from the program example shown in Figure 8.1 in Section 8.2.

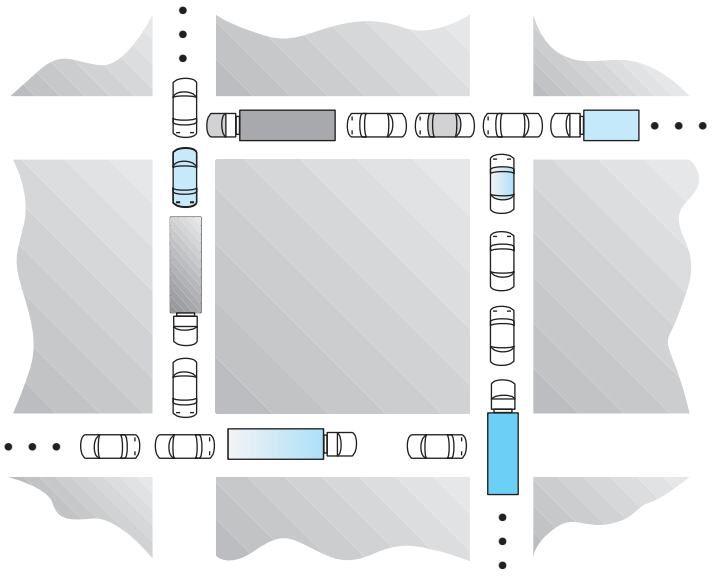

Figure 8.11 Traffic deadlock for Exercise 8.12.

8.14 In Section 6.8.1, we described a potential deadlock scenario involving processes P_0 and P_1 and semaphores `S` and `Q`. Draw the resource-allocation graph that illustrates deadlock under the scenario presented in that section.

8.15 Assume that a multithreaded application uses only reader–writer locks for synchronization. Applying the four necessary conditions for deadlock, is deadlock still possible if multiple reader–writer locks are used?

8.16 The program example shown in Figure 8.1 doesn't always lead to deadlock. Describe what role the CPU scheduler plays and how it can contribute to deadlock in this program.

8.17 In Section 8.5.4, we described a situation in which we prevent deadlock by ensuring that all locks are acquired in a certain order. However, we also point out that deadlock is possible in this situation if two threads simultaneously invoke the `transaction()` function. Fix the `transaction()` function to prevent deadlocks.

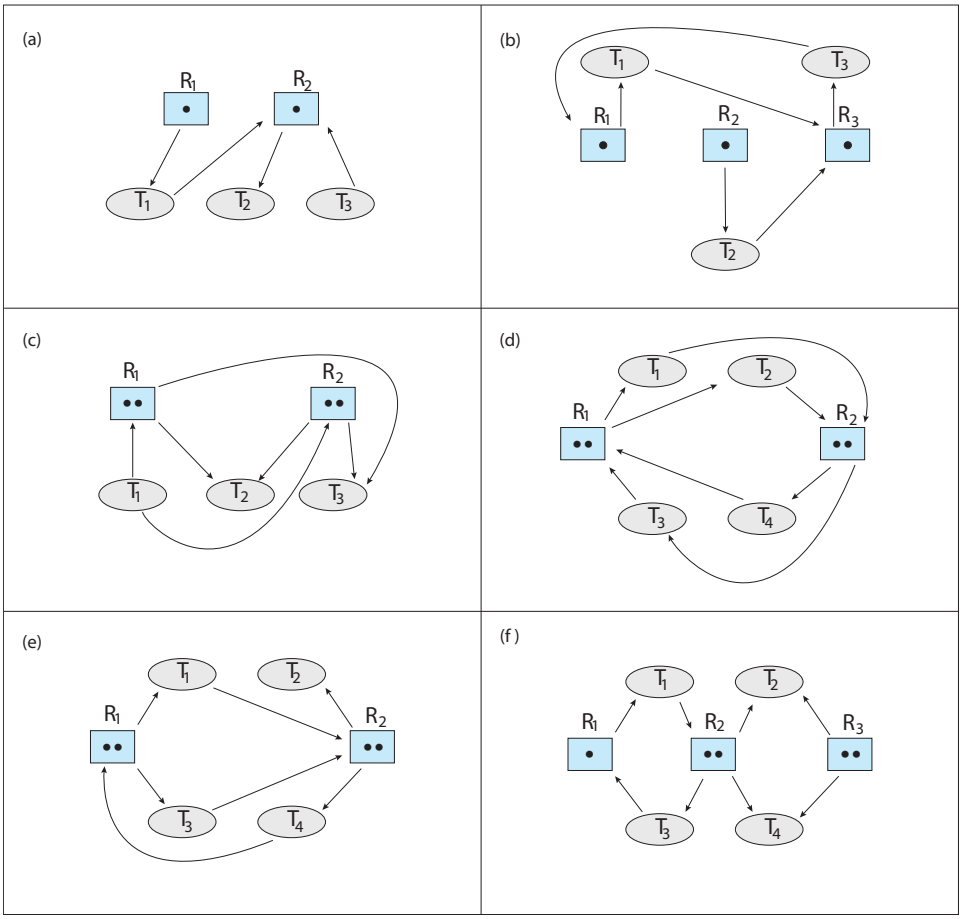

Figure 8.12 Resource-allocation graphs for Exercise 8.18.

8.18 Which of the six resource-allocation graphs shown in Figure 8.12 illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution.

8.19 Compare the circular-wait scheme with the various deadlock-avoidance schemes (like the banker's algorithm) with respect to the following issues:

a. Runtime overhead

b. System throughput

8.20 In a real computer system, neither the resources available nor the demands of threads for resources are consistent over long periods (months). Resources break or are replaced, new processes and threads come and go, and new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

a. Increase ***Available*** (new resources added).

b. Decrease ***Available*** (resource permanently removed from system).

c. Increase ***Max*** for one thread (the thread needs or wants more resources than allowed).

d. Decrease ***Max*** for one thread (the thread decides it does not need that many resources).

e. Increase the number of threads.

f. Decrease the number of threads.

8.21 Consider the following snapshot of a system:

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

What are the contents of the ***Need*** matrix?

8.22 Consider a system consisting of four resources of the same type that are shared by three threads, each of which needs at most two resources. Show that the system is deadlock free.

8.23 Consider a system consisting of m resources of the same type being shared by n threads. A thread can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold:

a. The maximum need of each thread is between one resource and m resources.

b. The sum of all maximum needs is less than $m + n$.

8.24 Consider the version of the dining-philosophers problem in which the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

8.25 Consider again the setting in the preceding exercise. Assume now that each philosopher requires three chopsticks to eat. Resource requests are still issued one at a time. Describe some simple rules for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

8.26 We can obtain the banker's algorithm for a single resource type from the general banker's algorithm simply by reducing the dimensionality of the various arrays by 1.

Show through an example that we cannot implement the multiple-resource-type banker's scheme by applying the single-resource-type scheme to each resource type individually.

8.27 Consider the following snapshot of a system:

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

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*** $= (2, 2, 2, 3)$

b. ***Available*** $= (4, 4, 1, 1)$

c. ***Available*** $= (3, 0, 1, 4)$

d. ***Available*** $= (1, 5, 2, 2)$

8.28 Consider the following snapshot of a system:

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

Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.

b. If a request from thread T_4 arrives for $(2, 2, 2, 4)$, can the request be granted immediately?

c. If a request from thread T_2 arrives for $(0, 1, 1, 0)$, can the request be granted immediately?

d. If a request from thread T_3 arrives for $(2, 2, 1, 2)$, can the request be granted immediately?

8.29 What is the optimistic assumption made in the deadlock-detection algorithm? How can this assumption be violated?

8.30 A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.) Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).

8.31 Modify your solution to Exercise 8.30 so that it is starvation-free.