**PROBLEMS**

**CHAPTER 6. DEADLOCKS**

1. 1. Give an example of a deadlock taken from politics.
2. 2. Students working at individual PCs in a computer laboratory send their files to be printed by a server that spools the files on its hard disk. Under what conditions may a deadlock occur if the disk space for the print spool is limited? How may the deadlock be avoided?
3. 3. In the preceding question, which resources are preemptable and which are nonpre-emptable?
4. 4. In Fig. 6-1 the resources are returned in the reverse order of their acquisition. Would giving them back in the other order be just as good?
5. 5. The four conditions (mutual exclusion, hold and wait, no preemption and circular wait) are necessary for a resource deadlock to occur. Give an example to show that these conditions are not sufficient for a resource deadlock to occur. When are these condi-tions sufficient for a resource deadock to occur?
6. 6. City streets are vulnerable to a circular blocking condition called gridlock, in which intersections are blocked by cars that then block cars behind them that then block the cars that are trying to enter the previous intersection, etc. All intersections around a city block are filled with vehicles that block the oncoming traffic in a circular manner.

Gridlock is a resource deadlock and a problem in competition synchronization. New York City’s prevention algorithm, called "don’t block the box," prohibits cars from entering an intersection unless the space following the intersection is also available. Which prevention algorithm is this? Can you provide any other prevention algorithms for gridlock?

1. 7. Suppose four cars each approach an intersection from four different directions simul-taneously. Each corner of the intersection has a stop sign. Assume that traffic regula-tions require that when two cars approach adjacent stop signs at the same time, the car on the left must yield to the car on the right. Thus, as four cars each drive up to their individual stop signs, each waits (indefinitely) for the car on the left to proceed. Is this anomaly a communication deadlock? Is it a resource deadlock?
2. 8. Is it possible that a resource deadlock involves multiple units of one type and a single unit of another? If so, give an example.
3. 9. Fig. 6-3 shows the concept of a resource graph. Do illegal graphs exist, that is, graphs that structurally violate the model we have used of resource usage? If so, give an ex-ample of one.
4. 10. Consider Fig. 6-4. Suppose that in step (o) *C* requested *S* instead of requesting *R*. Would this lead to deadlock? Suppose that it requested both *S* and *R*.
5. 11. Suppose that there is a resource deadlock in a system. Give an example to show that the set of processes deadlocked can include processes that are not in the circular chain in the corresponding resource allocation graph.
6. 12. In order to control traffic, a network router, *A* periodically sends a message to its neighbor, *B*, telling it to increase or decrease the number of packets that it can handle. At some point in time, Router *A* is flooded with traffic and sends *B* a message telling it to cease sending traffic. It does this by specifying that the number of bytes *B* may send (*A*’s window size) is 0. As traffic surges decrease, *A* sends a new message, telling *B* to restart transmission. It does this by increasing the window size from 0 to a positive number. That message is lost. As described, neither side will ever transmit. What type of deadlock is this?
7. 13. The discussion of the ostrich algorithm mentions the possibility of process-table slots or other system tables filling up. Can you suggest a way to enable a system administra-tor to recover from such a situation?
8. 14. Consider the following state of a system with four processes, *P1*, *P2*, *P3*, and *P4*, and five types of resources, *RS1*, *RS2*, *RS3*, *RS4*, and *RS5*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 1 | 1 | 2 |  | 1 | 1 | 0 | 2 | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| C = | 0 | 1 | 0 | 1 | 0 | R = | 0 | 1 | 0 | 2 | 1 | E = (24144) |
| 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 3 | 1 | A = (01021) |
|  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 | 1 | 0 | 0 | 0 |  | 0 | 2 | 1 | 1 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Using the deadlock detection algorithm described in Section 6.4.2, show that there is a deadlock in the system. Identify the processes that are deadlocked.

1. 15. Explain how the system can recover from the deadlock in previous problem using
   1. (a) recovery through preemption.
   2. (b) recovery through rollback.
   3. (c) recovery through killing processes.
2. 16. Suppose that in Fig. 6-6 *Cij* *Rij* > *E j* for some *i*. What implications does this have for the system?
3. 17. All the trajectories in Fig. 6-8 are horizontal or vertical. Can you envision any circum-stances in which diagonal trajectories are also possible?
4. 18. Can the resource trajectory scheme of Fig. 6-8 also be used to illustrate the problem of deadlocks with three processes and three resources? If so, how can this be done? If not, why not?
5. 19. In theory, resource trajectory graphs could be used to avoid deadlocks. By clever scheduling, the operating system could avoid unsafe regions. Is there a practical way of actually doing this?
6. 20. Can a system be in a state that is neither deadlocked nor safe? If so, give an example. If not, prove that all states are either deadlocked or safe.
7. 21. Take a careful look at Fig. 6-11(b). If *D* asks for one more unit, does this lead to a safe state or an unsafe one? What if the request came from *C* instead of *D* ?
8. 22. A system has two processes and three identical resources. Each process needs a maxi-mum of two resources. Is deadlock possible? Explain your answer.
9. 23. Consider the previous problem again, but now with *p* processes each needing a maxi-mum of *m* resources and a total of *r* resources available. What condition must hold to make the system deadlock free?
10. 24. Suppose that process *A* in Fig. 6-12 requests the last tape drive. Does this action lead to a deadlock?
11. 25. The banker’s algorithm is being run in a system with *m* resource classes and *n* proc-esses. In the limit of large *m* and *n*, the number of operations that must be performed to check a state for safety is proportional to *ma nb*. What are the values of *a* and *b*?
12. 26. A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Allocated* | *Maximum* | *Available* |
| Process A | 10211 | 11213 | 0 0 x 1 1 |
| Process B | 20110 | 22210 |  |
| Process C | 11010 | 21310 |  |
| Process D | 11110 | 11221 |  |

What is the smallest value of *x* for which this is a safe state?

1. 27. One way to eliminate circular wait is to have rule saying that a process is entitled only to a single resource at any moment. Give an example to show that this restriction is unacceptable in many cases.
2. 28. Two processes, *A* and *B*, each need three records, 1, 2, and 3, in a database. If *A* asks for them in the order 1, 2, 3, and *B* asks for them in the same order, deadlock is not possible. However, if *B* asks for them in the order 3, 2, 1, then deadlock is possible. With three resources, there are 3! or six possible combinations in which each process can request them. What fraction of all the combinations is guaranteed to be deadlock free?
3. 29. A distributed system using mailboxes has two IPC primitives, send and receive. The latter primitive specifies a process to receive from and blocks if no message from that process is available, even though messages may be waiting from other processes. There are no shared resources, but processes need to communicate frequently about other matters. Is deadlock possible? Discuss.
4. 30. In an electronic funds transfer system, there are hundreds of identical processes that work as follows. Each process reads an input line specifying an amount of money, the account to be credited, and the account to be debited. Then it locks both accounts and transfers the money, releasing the locks when done. With many processes running in parallel, there is a very real danger that a process having locked account *x* will be unable to lock *y* because *y* has been locked by a process now waiting for *x.* Devise a scheme that avoids deadlocks. Do not release an account record until you have com-pleted the transactions. (In other words, solutions that lock one account and then re-lease it immediately if the other is locked are not allowed.)
5. 31. One way to prevent deadlocks is to eliminate the hold-and-wait condition. In the text it was proposed that before asking for a new resource, a process must first release what-ever resources it already holds (assuming that is possible). However, doing so intro-duces the danger that it may get the new resource but lose some of the existing ones to competing processes. Propose an improvement to this scheme.
6. 32. A computer science student assigned to work on deadlocks thinks of the following bril-liant way to eliminate deadlocks. When a process requests a resource, it specifies a time limit. If the process blocks because the resource is not available, a timer is start-ed. If the time limit is exceeded, the process is released and allowed to run again. If you were the professor, what grade would you give this proposal and why?
7. 33. Main memory units are preempted in swapping and virtual memory systems. The processor is preempted in time-sharing environments. Do you think that these preemp-tion methods were developed to handle resource deadlock or for other purposes? How high is their overhead?
8. 34. Explain the differences between deadlock, livelock, and starvation.
9. 35. Assume two processes are issuing a seek command to reposition the mechanism to ac-cess the disk and enable a read command. Each process is interrupted before executing its read, and discovers that the other has moved the disk arm. Each then reissues the seek command, but is again interrupted by the other. This sequence continually repeats. Is this a resource deadlock or a livelock? What methods would you recommend to handle the anomaly?
10. 36. Local Area Networks utilize a media access method called CSMA/CD, in which sta-tions sharing a bus can sense the medium and detect transmissions as well as collisions. In the Ethernet protocol, stations requesting the shared channel do not transmit frames if they sense the medium is busy. When such transmission has terminated, waiting stations each transmit their frames. Two frames that are transmitted at the same time will collide. If stations immediately and repeatedly retransmit after collision de-tection, they will continue to collide indefinitely.
    1. (a) Is this a resource deadlock or a livelock?
    2. (b) Can you suggest a solution to this anomaly?
    3. (c) Can starvation occur with this scenario?
11. 37. A program contains an error in the order of cooperation and competition mechanisms, resulting in a consumer process locking a mutex (mutual exclusion semaphore) before it blocks on an empty buffer. The producer process blocks on the mutex before it can place a value in the empty buffer and awaken the consumer. Thus, both processes are blocked forever, the producer waiting for the mutex to be unlocked and the consumer waiting for a signal from the producer. Is this a resource deadlock or a communication deadlock? Suggest methods for its control.
12. 38. Cinderella and the Prince are getting divorced. To divide their property, they have agreed on the following algorithm. Every morning, each one may send a letter to the other’s lawyer requesting one item of property. Since it takes a day for letters to be de-livered, they have agreed that if both discover that they have requested the same item on the same day, the next day they will send a letter canceling the request. Among their property is their dog, Woofer, Woofer’s doghouse, their canary, Tweeter, and Tweeter’s cage. The animals love their houses, so it has been agreed that any division of property separating an animal from its house is invalid, requiring the whole division to start over from scratch. Both Cinderella and the Prince desperately want Woofer. So that they can go on (separate) vacations, each spouse has programmed a personal com-puter to handle the negotiation. When they come back from vacation, the computers are still negotiating. Why? Is deadlock possible? Is starvation possible? Discuss your answer.
13. 39. A student majoring in anthropology and minoring in computer science has embarked on a research project to see if African baboons can be taught about deadlocks. He locates a deep canyon and fastens a rope across it, so the baboons can cross hand-over-hand. Several baboons can cross at the same time, provided that they are all going in the same direction. If eastward-moving and westward-moving baboons ever get onto the rope at the same time, a deadlock will result (the baboons will get stuck in the mid-dle) because it is impossible for one baboon to climb over another one while suspended over the canyon. If a baboon wants to cross the canyon, he must check to see that no other baboon is currently crossing in the opposite direction. Write a program using semaphores that avoids deadlock. Do not worry about a series of eastward-moving baboons holding up the westward-moving baboons indefinitely.
14. 40. Repeat the previous problem, but now avoid starvation. When a baboon that wants to cross to the east arrives at the rope and finds baboons crossing to the west, he waits until the rope is empty, but no more westward-moving baboons are allowed to start until at least one baboon has crossed the other way.
15. 41. Program a simulation of the banker’s algorithm. Your program should cycle through each of the bank clients asking for a request and evaluating whether it is safe or unsafe. Output a log of requests and decisions to a file.
16. 42. Write a program to implement the deadlock detection algorithm with multiple re-sources of each type. Your program should read from a file the following inputs: the number of processes, the number of resource types, the number of resources of each type in existence (vector *E*), the current allocation matrix *C* (first row, followed by the second row, and so on), the request matrix *R* (first row, followed by the second row, and so on). The output of your program should indicate whether there is a deadlock in the system. In case there is, the program should print out the identities of all processes that are deadlocked.
17. 43. Write a program that detects if there is a deadlock in the system by using a resource al-location graph. Your program should read from a file the following inputs: the number of processes and the number of resources. For each process if should read four num-bers: the number of resources it is currently holding, the IDs of resources it is holding, the number of resources it is currently requesting, the IDs of resources it is requesting. The output of program should indicate if there is a deadlock in the system. In case there is, the program should print out the identities of all processes that are deadlocked.
18. 44. In certain countries, when two people meet they bow to each other. The protocol is that one of them bows first and stays down until the other one bows. If they bow at the same time, they will both stay bowed forever. Write a program that does not deadlock.