

# **Chapter 5**

## **Process Management**

### **At a Glance**

#### **Instructor's Manual Table of Contents**

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

## Lecture Notes

### Overview

This chapter presents and discusses problems that arise when many processes compete for relatively few resources and the system is unable to service all of the processes. Specifically, two extreme conditions, deadlock and starvation, result from a lack of process synchronization. This chapter begins by discussing different types of deadlocks, followed by a discussion of the conditions causing deadlock. A presentation on deadlock modeling using directed graphs is provided, and an explanation of the strategies for dealing with deadlock follows. Finally, a detailed discussion on starvation is presented.

### Learning Objectives

After completing this chapter, the student should be able to describe:

- Several causes of system deadlock
- The difference between preventing and avoiding deadlocks
- How to detect and recover from deadlocks
- The concept of process starvation and how to detect and recover from it
- The concept of a race and how to prevent it
- The difference between deadlock, starvation, and race

### Teaching Tips

#### **Deadlock**

1. Review the two aspects of resource sharing discussed in earlier chapters of the text: memory management and processor sharing. Inform the students that when many processes compete for relatively few resources, problems may arise if the system is unable to service all of the processes in a synchronous manner.
2. Introduce students to the concept of process synchronization. Explain how a lack of process synchronization results in two extreme conditions in the system known as deadlock or starvation. Describe the term deadlock by reviewing its predecessor descriptive phrase known as “deadly embrace.” Review the staircase example on page 142 of the text to provide students with a better understanding of deadlock using nontechnical concepts.
3. Point out that deadlock is more serious than starvation because it affects more than one job. Discuss how the issue of resources being tied up or unavailable creates a problem for the entire system, not just a few programs.

4. Clarify this point pictorially using the traffic jam example illustrated in Figure 5.1 on page 143 of the text. Emphasize that with any deadlock situation, there is no simple and immediate solution and outside intervention is necessary to resolve it.
5. Point out that the resource usage in today's interactive systems has led to deadlock increases. Furthermore, a real-time environment experiencing a deadlock occurrence results in a critical situation. Provide examples of the impact of deadlock on critical real-time situations, such as those described on page 144 in the text.

<b><i>Teaching Tip</i></b>	Compared to older batch oriented systems, today's interactive and real-time systems are quite flexible and improve resource usage through dynamic resource sharing. These features have contributed to an increased prevalence of the deadlock problem.
----------------------------	---

## Seven Cases of Deadlock

1. Explain that deadlocks often result when nonsharable and nonpreemptable resources are allocated to jobs that eventually require other nonsharable and nonpreemptable resources that have been locked and made unavailable.
2. Outline the seven cases of deadlock that will be discussed in this section.

<b><i>Teaching Tip</i></b>	Mention that deadlocks are not restricted to files, printers, and tape drives. They may also occur on sharable resources such as disks and databases.
----------------------------	---

### Case 1: Deadlocks on File Requests

1. Explain that deadlock from multiple process file requests occurs because of jobs being allowed to hold files for the duration of their executions.
2. Use the example shown in Figure 5.2 on page 144 in the text to demonstrate a file request deadlock occurrence and the sequence of steps preceding the situation. Point out that the deadlock will remain until one of the two programs is withdrawn or forcibly removed and its file is released.

### Case 2: Deadlocks in Databases

1. Explain that deadlock may occur if two processes access and lock records in a database. Remind students that database queries and transactions are often relatively brief and random.

2. Explain the concept of locking within a database whereby one user locks out all other users while working with the database. The technique is used to guarantee the integrity of the data. Outline the three levels at which locking is implemented.
3. Use the generic example provided on page 145 in the text to demonstrate a database record request deadlock occurrence and the sequence of steps preceding the situation.
4. Use the example shown in Figure 5.3 on page 146 in the text to demonstrate how the condition of “race between processes” is reached when locking is not used. Discuss its consequences.

<b>Teaching Tip</b>	Race between processes is an important concept. Ensure that students understand what it is, what causes it, and how it differs from deadlock and starvation.
---------------------	--

### **Case 3: Deadlocks in Dedicated Device Allocation**

1. Explain that deadlock may occur if there are a limited number of dedicated devices and many more processes requesting them.
2. Use the generic example provided on page 147 in the text to demonstrate how two processes, requesting exclusive use of two tape drives each, may result in a deadlock situation. Note that in this scenario, there are only two tape drives available in the system for all processes.

### **Case 4: Deadlocks in Multiple Device Allocation**

1. Explain that deadlock may occur with multiple device allocations utilizing different device types when several processes request and hold dedicated devices while other processes act in a similar manner.
2. Use the example shown in Figure 5.4 on page 147 in the text to demonstrate the case of deadlocks in multiple device allocation.

### **Case 5: Deadlocks in Spooling**

1. Explain the concepts of virtual devices (sharable devices) and spooling.
2. Discuss how deadlock may occur if a sharable device, such as a spooling printer that requires all job output before initiating printing, encounters a full spool disk area with only partially completed output residing on it.
3. Use the generic example provided on page 148 in the text to demonstrate how a spool disk area may become full prior to any one job having its entire print output in the spool area. Thus, the printer cannot print any output, and no more output may be added to the spool area for printing resulting in deadlock.

**Teaching  
Tip**

Note that deadlock in spooling is not limited to printers. Any part of the system that relies on spooling is vulnerable to a similar scenario.

**Case 6: Deadlocks in a Network**

1. Discuss the case of deadlock in a network resulting from the absence of network protocols to control message flow.
2. Use the example shown in Figure 5.5 on page 149 of the text to illustrate how a network that is congested or one that has filled a large percentage of its I/O buffer space becomes deadlocked.

**Case 7: Deadlocks in Disk Sharing**

1. Discuss the case of deadlock resulting from disk sharing.
2. Use the example shown in Figure 5.6 on page 150 of the text to illustrate how livelock results from competing processes by sending conflicting commands to access different areas on the same disk.

**Teaching  
Tip**

Refer to the following Web site to learn about deadlock in networks:  
[www.research.ibm.com/journal/rd/231/ibmrd2301K.pdf](http://www.research.ibm.com/journal/rd/231/ibmrd2301K.pdf)

**Conditions for Deadlock**

1. Mention the four conditions that need to be simultaneously present for deadlock to occur: mutual exclusion, resource holding, no preemption, and circular wait.
2. Be sure to emphasize that each of these four individual conditions is necessary for the operating system to work smoothly, and none of them can be removed easily without causing the system's overall functioning to suffer. Therefore, the system needs to recognize the combination of conditions before they occur and threaten to cause the system to lock up.
3. Explain briefly each of the four conditions. Use the examples with the staircase scenario as mentioned in the text on page 151 to illustrate these conditions. Be certain students understand that removal of only one condition can resolve the deadlock.

## **Quick Quiz 1**

1. Which of the following conditions results from the liberal allocation of resources?
  - a. Starvation
  - b. Deadlock
  - c. Increased efficiency
  - d. Request for more memoryAnswer: b
  
2. Which of the following conditions are necessary for deadlock to occur? (Choose all that apply.)
  - a. Mutual exclusion
  - b. Preemption
  - c. Circular wait
  - d. Process holdingAnswer: a and c
  
3. \_\_\_\_\_ is the act of allowing only one process to have access to a dedicated resource.  
Answer: Mutual exclusion

## **Modeling Deadlocks**

1. Explain directed graphs and how they are used to model the four simultaneous conditions necessary for deadlock. Make sure the students pay attention to the arrow direction, as it determines whether the process is holding the resource or waiting for a resource. Emphasize that a cycle in the graph indicates a deadlock condition. Use the example shown in Figure 5.7 on page 152 of the text to illustrate directed graphs.
2. Explain Scenario One, which demonstrates how deadlock cannot occur even if each process requests every resource *if* the resources are released before the next process requests them. Use Table 5.1 and Figure 5.8 on page 152 of the text to illustrate this scenario.
3. Explain Scenario Two, which demonstrates how deadlock occurs when every process is waiting for a resource being held by another process, but none is released without operator intervention. Use Table 5.2 and Figure 5.9 on page 153 of the text to illustrate this scenario.
4. Explain Scenario Three, which demonstrates how deadlock is avoided by releasing resources before the deadlock can occur. Use Table 5.3 and Figure 5.10 on pages 153 and 154 of the text to illustrate this scenario.
5. Point out that directed graphs can cluster devices of the same type into one entity, represented by a rectangle. Use the example shown in Figure 5.11 on page 155 of the text to illustrate clustering.

## Strategies for Handling Deadlocks

1. Outline different strategies that operating systems use to deal with deadlocks (prevention, avoidance, and detection).

### Prevention

1. Describe the strategy of preventing deadlock in which it is necessary to eliminate one of the four necessary simultaneous conditions present in a deadlock situation. Note that this task is complicated by the fact that the same condition cannot be eliminated from every resource.
2. Explain the difficulties in eliminating mutual exclusion; it is necessary in any computer system. Describe how mutual exclusion may be bypassed by spooling; however, this may lead to another type of deadlock situation from the spooling function.
3. Explain the difficulties in eliminating resource holding and the resulting consequences. The consequences involve a potential decrease in multiprogramming and idle devices within the system. Note that resource holding may be avoided by forcing each job to request, at creation time, every resource it will need. Unfortunately, this significantly decreases the degree of multiprogramming.
4. Explain the difficulties in eliminating no preemption. Note that it is not acceptable to preempt dedicated I/O devices or files during modification. Also note that no preemption could be bypassed by allowing the operating system to deallocate resources from jobs. This tactic only works well if the job state is easily saved and restored.
5. Explain the difficulties in eliminating a circular wait. Note that it can be bypassed if the operating system prevents the formation of a circle. One solution is a numbering system for the resources; however, it requires jobs to anticipate the order in which they will request resources. Explain how this works by using the example in the text on page 156. Another solution is the scheme of hierarchical ordering. Explain the advantages and disadvantages of this scheme, as discussed in the text on page 156.

### Avoidance

1. Explain to students that prevention may not be possible. In such a case, avoidance of the deadlock situation should be attempted. Point out that deadlock can be avoided if the system knows ahead of time the sequence of requests associated with each of the active processes.
2. Explain Dijkstra's Bankers Algorithm (1965) that regulates resource allocation to avoid deadlock. Use Table 5.4 through Table 5.7 on pages 157-158 of the text to illustrate this algorithm.

3. Discuss “safe” and “unsafe” states. Outline various factors that an operating system should consider to avoid deadlock. For example, it should never satisfy a request that moves it from a safe state to an unsafe one. Also, it must identify the job with the smallest number of remaining resources. Finally, it must make sure that the number of available resources is always equal to, or greater than, the number needed for the selected job.
4. Outline several difficulties with the Banker’s Algorithm, as listed in the bullet points on page 159 in the text.

### Detection

1. Explain that should prevention and avoidance fail, then the operating system should use the detection algorithm. Describe how it is used to detect circularity in order to remove deadlock from the system. Review the algorithm on pages 159-160 of the text.
2. Use the example shown in Figures 5.12 and 5.13 on pages 160 and 161 of the text to illustrate this algorithm.

### Recovery

1. Explain that once a deadlock is detected, it must be untangled and the system returned to normal as soon as possible. Discuss the six recovery methods as explained on page 162 of the text. Emphasize that all these methods have a common feature of requiring at least one victim, and unfortunately, the victim removal generally requires that the job be restarted from the beginning or from a convenient midpoint.
2. Outline different factors that must be considered to select the victim that will have the least negative effect on the system. The most common include: (1) the priority of the job under consideration, that is, high-priority are jobs usually untouched, (2) CPU time used by job, that is, jobs close to completion are usually left alone, (3) the number of other jobs that would be affected if this job were selected as the victim. In addition, jobs that are modifying data are usually not selected for termination because the consistency and validity of the data would be jeopardized (a database issue).

<b>Teaching Tip</b>	Refer to the following Web site to learn more about operating systems algorithms: <a href="http://www.darkridge.com/~jpr5/archive/alg/node148.html">www.darkridge.com/~jpr5/archive/alg/node148.html</a>
---------------------	---

### Starvation

1. Provide the students with an overview of starvation. Point out that deadlock results from the liberal allocation of resources, while starvation results from conservative allocation of resources. Students should have a clear understanding of the difference between deadlock and starvation.

2. Use the example of “The dining philosophers,” as shown in Figures 5.14 and 5.15 on pages 163 and 164 of the text, to illustrate starvation.
3. Explain how starvation can be avoided. Solutions include implementing an algorithm to track how long each job has been waiting for resources (aging), and blocking new jobs until the starving jobs have been satisfied.

## **Quick Quiz 2**

1. In directed graphs, a solid arrow from a resource to process represents which of the following?
  - a. Process is holding the resource
  - b. Process is waiting for the resource
  - c. Deadlock
  - d. StarvationAnswer: a
2. Which of the following strategies for handling deadlocks is illustrated using the Banker’s algorithm?
  - a. Prevention
  - b. Avoidance
  - c. Detection
  - d. RecoveryAnswer: b
3. Which of the following methods are used for recovery? (Choose all that apply.)
  - a. Terminate every job that is active in the system and restart them from the beginning
  - b. Allow new jobs to enter the system
  - c. Terminate only the jobs involved in the deadlock and ask their users to resubmit them
  - d. Select a nondeadlocked job, preempt the resources it is holding, and allocate them to a deadlocked processAnswer: a, c, and d

## **Class Discussion Topics**

1. Have students discuss the necessary conditions for a deadlock to occur. If given an option of removing one condition in order to prevent deadlock, which condition would they eliminate? Why?
2. Have students discuss different deadlock handling strategies. Which strategies would they like to implement to remove deadlocks in the cases of disk sharing, database sharing, and multiple device allocation? Have students select a suitable strategy in each case and provide reasons for their choices.

## Additional Projects

1. Have students compile a list of algorithms employed in today's operating systems to avoid deadlocks.
2. Have students research an algorithm to check network states for a deadlock.

## Additional Resources

1. Intel.com:  
[www.intel.com](http://www.intel.com)
2. Freescale.com:  
[www.freescale.com](http://www.freescale.com)
3. IBM.com:  
[www.ibm.com](http://www.ibm.com)
4. Microsoft.com:  
[www.microsoft.com](http://www.microsoft.com)
5. Osddata.com:  
[www.osdata.com/kind/history.htm](http://www.osdata.com/kind/history.htm)

## Key Terms

- **Avoidance:** the dynamic strategy of deadlock avoidance that attempts to ensure that resources are never allocated in such a way as to place a system in an unsafe state.
- **Circular wait:** one of four conditions for deadlock through which each process involved is waiting for a resource being held by another; each process is blocked and can't continue, resulting in deadlock.
- **Deadlock:** a problem occurring when the resources needed by some jobs to finish execution are held by other jobs, which, in turn, are waiting for other resources to become available. Also called deadly embrace.
- **Detection:** the process of examining the state of an operating system to determine whether a deadlock exists.
- **Directed graphs:** a graphic model representing various states of resource allocations.
- **Livelock:** a locked system whereby two (or more) processes continually block the forward progress of the other without making any forward progress itself. It is similar to a deadlock except that neither process is blocked or waiting; both are in a continuous state of change.
- **Locking:** a technique used to guarantee the integrity of the data in a database through which the user locks out all other users while working with the database.
- **Mutual exclusion:** one of four conditions for deadlock in which only one process is allowed to have access to a resource.

- **No preemption:** one of four conditions for deadlock in which a process is allowed to hold on to resources while it is waiting for other resources to finish execution.
- **Prevention:** a design strategy for an operating system where resources are managed in such a way that some of the necessary conditions for deadlock do not hold.
- **Process synchronization:** (1) the need for algorithms to resolve conflicts between processors in a multiprocessing environment; or (2) the need to ensure that events occur in the proper order even if they are carried out by several processes.
- **Race:** a synchronization problem between two processes vying for the same resource.
- **Recovery:** the steps that must be taken, when deadlock is detected, by breaking the circle of waiting processes.
- **Resource holding:** one of four conditions for deadlock in which each process refuses to relinquish the resources it holds until its execution is completed even though it isn't using them because it's waiting for other resources.
- **Safe state:** the situation in which the system has enough available resources to guarantee the completion of at least one job running on the system.
- **Spooling:** a technique developed to speed I/O by collecting in a disk file either input received from slow input devices or output going to slow output devices, such as printers.
- **Starvation:** the result of conservative allocation of resources in which a single job is prevented from execution because it's kept waiting for resources that never become available.
- **Unsafe state:** a situation in which the system has too few available resources to guarantee the completion of at least one job running on the system. It can lead to deadlock.
- **Victim:** an expendable job that is selected for removal from a deadlocked system to provide more resources to the waiting jobs and resolve the deadlock.