

# Chapter 6

## Concurrent Processes

### 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 looks at multiprocessing systems where there are several processors working together in a variety of configurations. The chapter begins with a detailed explanation of parallel processing. The evolution of multiprocessors is presented next, followed by an introduction to multi-core processors. This discussion is followed by an examination of the three typical multiprocessing configurations found in systems today: master/slave, loosely coupled, and symmetric. A presentation on the role of process synchronization software including the specifics of how synchronization is sometimes implemented as a lock-and-key arrangement follows. The chapter continues with a discussion on process cooperation and reviews the classic problems of concurrent processes, such as producers and consumers, readers and writers, and the missed waiting customer. The issues of concurrent programming and threads in a multiprocessing environment are discussed. The chapter concludes with a brief look at concurrent programming languages, concentrating on the Java programming language, the Java platform, and the Java language environment.

### Chapter Objectives

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

- The critical difference between processes and processors, and their connection
- The differences among common configurations of multiprocessing systems
- The basic concepts of multi-core processor technology
- The significance of a critical region in process synchronization
- The essential ideas behind process synchronization software
- The need for process cooperation when several processors work together
- The similarities and differences between processes and threads
- How processors cooperate when executing a job, process, or thread
- The significance of concurrent programming languages and their applications

### Teaching Tips

#### **What Is Parallel Processing?**

1. Provide students with an overview of the concept of parallel processing. Note that parallel processing is also referred to as multiprocessing. Discuss how parallel processing involves two or more processors operating in unison, which in turn suggests two or more CPUs executing instructions simultaneously.
2. Outline the challenges the Processor Manager faces when dealing with multiprocessors, such as a need to coordinate the activity of each processor and a need to synchronize the interaction among the CPUs.

3. Identify the reasons for the development of parallel processing, including solving the need to enhance throughput and the need to increase computing power. Outline the benefits of parallel processing, such as increased reliability and faster processing. Note that increased reliability results from having more than one CPU available. That is, if one processor fails, the others take over, but this is not an easy task to implement. Also note that faster processing results from instructions being processed two or more at a time in parallel.
4. Expand upon parallel instruction processing by describing the three scenarios in which it occurs. These include: (1) allocating a CPU to each program or job, (2) allocating a CPU to each working set or parts of it, (3) subdividing individual instructions so each subdivision can be processed simultaneously (concurrent programming).
5. Point out two major challenges involved in parallel processing: (1) how to connect the processors into configurations and (2) how to orchestrate their interaction.
6. Illustrate the complexities of the Processor Manager's task when dealing with multiple processors or multiple processes using the high-level fast-food example on page 174 in the text. Review the six-step information retrieval process described in Table 6.1 on page 175 in the text.

<b><i>Teaching Tip</i></b>	Be sure to emphasize to students that synchronization is the key to success in multiprocessor systems, and the system cannot work properly unless every processor communicates and cooperates with every other processor.
----------------------------	---

## Evolution of Multiprocessors

1. Note the history of multiprocessing; it was developed for high-end midrange and mainframe computers, and each additional CPU was treated as an additional resource that could schedule work. As CPU hardware costs declined (in the 1980s), the availability of multiprocessing expanded; today, multiprocessing systems are available on systems of every size.
2. Use Table 6.2 on page 176 in the text to summarize the three levels where multiprocessing may occur.

## Introduction to Multi-Core Processors

1. Discuss the hardware innovation of the multi-core processor whereby several processors are placed on a single chip. Discuss the problems of current leakage (tunneling) and heat.

2. Explain the solution resolving the current leak and heat problems. Emphasize that the solution involves designing multiple processor cores on a single chip that allow two (or more) sets of calculations to take place at the same time. The two (or more) cores generate less heat than a single processor of the same size and current leakage (tunneling) is reduced.
3. Point out that a disadvantage of this processor is that the two cores run more slowly than single processor chips. As discussed on page 177 of the text, to gain performance from a dual-core chip, software has to be structured to take advantage of the double calculation capability.

<b>Teaching Tip</b>	Note how placing two processor cores on a single chip affects the operating system because the system must now manage multiple processors, multiple RAMs, and the processing of many tasks at once.
<b>Teaching Tip</b>	Find out more on Moore's Law here: <a href="http://www.intel.com/technology/mooreslaw/index.htm">www.intel.com/technology/mooreslaw/index.htm</a>

## Typical Multiprocessing Configurations

1. Note that much depends on how the multiple processors are configured within a system. Outline typical multiprocessing configurations, such as master/slave, loosely coupled, and symmetric.

### Master/Slave Configuration

1. Explain the master/slave multiprocessing configuration. Outline the responsibilities of the master processor. Discuss the computing environments for which this type of configuration is suitable.
2. Use Figure 6.1 on page 178 in the text to illustrate the concept.
3. Outline the major advantages and disadvantages of this configuration. Point out that if the master processor fails, the entire system fails. This configuration also increases the number of interrupts because all slave processors must interrupt the master processor every time they need operating system intervention, such as for I/O requests.

### Loosely Coupled Configuration

1. Explain the loosely coupled multiprocessing configuration. Point out to students that each processor controls its own resources.
2. Use Figure 6.2 on page 178 in the text to illustrate the concept.

3. Discuss the only difference between a loosely coupled systems and a collection of independent single-processing systems. Explain the need for global tables by each processor.
4. Point out that job scheduling is based on several requirements and policies. Indicate the advantages and disadvantages of the configuration. The system is not prone to catastrophic system failures; however, it is difficult to detect when a processor has failed.

### Symmetric Configuration

1. Explain the symmetric multiprocessing configuration. Outline its advantages over the loosely coupled configuration, such as: (1) more reliable; (2) uses resources effectively; (3) can balance loads well; and (4) can degrade gracefully in the event of a failure. Point out that this is the most difficult configuration to implement because the processes must be well synchronized to avoid the problems of races and deadlocks.
2. Use Figure 6.3 on page 179 in the text to illustrate this configuration. Note the single copy of the operating system and the use of a multiple processor. This requires the use of a global table listing. Describe the processor procedure in this configuration when a process is interrupted.
3. Note the need for process synchronization in this configuration.

<b>Teaching Tip</b>	For information on a large symmetric IBM multiprocessing system, refer to the article at: <a href="http://www.research.ibm.com/journal/rd/483/mak.html">www.research.ibm.com/journal/rd/483/mak.html</a>
---------------------	--

### Process Synchronization Software

1. Discuss the requirements for successful process synchronization. Specifically, a used resource must be locked away from other processes until it is released, and a waiting process is allowed to use that resource only when it is released. Mention that a mistake in this area could leave a job waiting indefinitely or cause a deadlock if the resource is a key resource.
2. Illustrate this point using the ice cream shop and processor examples provided on pages 180 and 181 in the text.
3. Emphasize the importance of synchronization. Explain the concept of the critical region, which is a part of a program that must complete execution before other processes can have access to the resources being used. Be sure to mention that processes within a critical region cannot be interleaved without threatening the integrity of the operation.

4. Explain how synchronization is similar to a lock-and-key arrangement. Outline how the synchronization sequence works. First, the process must first see if the key is available. Second, if it is available, the process must pick it up and put it in the lock to make it unavailable to all other processes. Explain why both actions must be performed in a single machine cycle.
5. Mention several locking mechanisms. These include test-and-set, WAIT and SIGNAL, and semaphores.

### **Test-and-Set**

1. Explain the test-and-set mechanism, in which an indivisible machine instruction is executed in a single machine cycle to see if the key is available. If so, its status is set to unavailable.
2. Discuss the actual key in terms of construction, use, and storage.
3. Point out that it is a simple procedure to implement and works well for a small number of processes. Discuss the major drawbacks of this mechanism: (1) starvation could occur when many processes are waiting to enter a critical region, and (2) waiting processes remain in unproductive, resource-consuming wait loops (busy waiting).

### **WAIT and SIGNAL**

1. Explain the WAIT and SIGNAL mechanism, a modification of test-and-set that is designed to remove the drawback of busy waiting. Discuss two new mutually exclusive operations, WAIT and SIGNAL, which are part of the Process Scheduler's set of operations.

### **Semaphores**

1. Explain the mechanism of semaphores. Use the railroad example and Figure 6.4 on page 183 in the text to illustrate this concept.
2. Explain the P and V operations to operate the semaphore to overcome the process synchronization problem. Point out that P stands for the Dutch word *proberen* (to test) and V stands for *verhogen* (to increment).
3. Use Table 6.3 on page 184 along with the discussion on page 183 in the text to clarify these operations. Be sure to point out that in the case of multiple waiting jobs, the next job to be processed is determined by the algorithm used by this portion of the Process Scheduler.
4. Point out that the P and V operations depend on a semaphore to enforce the concept of mutual exclusion and that such a semaphore is called a mutex (MUTual EXclusion).

5. Stress that the problem of mutual exclusion may apply to several processes on more than one processor, or interacting (codependent) processes on a single processor. In this case, the concept of a critical region becomes necessary because it ensures that parallel processes will modify shared data only while in the critical region.
6. Explain that in sequential computations, mutual exclusion is achieved automatically. In parallel computations, the order of execution can change, so mutual exclusion must be explicitly stated and maintained.

<b>Teaching Tip</b>	Refer to the following Web site to download a detailed article on semaphore operations: <a href="http://docs.hp.com/en/TKP-90203/ch05s04.html">http://docs.hp.com/en/TKP-90203/ch05s04.html</a>
---------------------	---

## Process Cooperation

1. Provide a brief overview of the concept of process cooperation, and discuss the problems that may arise in the absence of mutual exclusion and synchronization. Discuss two common examples: problems of producers and consumers, and of readers and writers. Mention that each case requires both mutual exclusion and synchronization, and each is implemented by using semaphores.

### Producers and Consumers

1. Discuss the problem of producers and consumers, which arises when one process produces some data that another process consumes later. Present the “cook and bagger” example as mentioned on page 185 in the text to illustrate this problem in non-technical terms.
2. Explain this problem in relation to a computer system using the example of a buffer between a CPU and a line printer. Explain how the buffer is used to synchronize the process between the CPU and a line printer. Use Figure 6.5 on page 186 in the text to illustrate the buffer states and actions taken for each.
3. Explain how this problem can be solved using the concept of semaphores. Explain the definitions of producer and consumer processes using Tables 6.4 and 6.5 in the text on pages 186 and 187.
4. Explain the Producers and Consumers Algorithm. Make sure students clearly understand the definitions of the variables and functions used in the Producers and Consumers Algorithm.

### Readers and Writers

1. Discuss the problem of readers and writers, which arises when two types of processes need to access a shared resource such as a file or database. Use the example of an airline reservation system as mentioned on page 187 in the text to illustrate this problem.

2. Explain how this problem can be solved by using the concept of semaphores.

## **Quick Quiz 1**

1. Which of the following multiprocessing configurations is least reliable?
  - a. Master/slave
  - b. Loosely coupled
  - c. Tightly coupled
  - d. Symmetric

Answer: a

2. Which of the following conditions of semaphore variable “s” implies a busy critical region?
  - a.  $s > 0$
  - b.  $s < 0$
  - c.  $s = 0$
  - d.  $s > 1$

Answer: c

3. The common element in all synchronization schemes is to allow a process to finish work on a(n) \_\_\_\_\_ of the program before other processes have access to it.

Answer: critical region

## **Concurrent Programming**

1. Provide an overview of the concept of a concurrent processing system. Note that in this case, multiprocessing refers to one job using several processors to execute sets of instructions in parallel and that it requires a programming language and system to support it.

## **Applications of Concurrent Programming**

1. Discuss how concurrent programming is useful in solving arithmetic equations. Discuss advantages of programming languages that are serial in nature, using the examples shown in Tables 6.2 and 6.3 on pages 189 and 190 in the text.
2. Discuss the concept of explicit parallelism, which requires that the programmer explicitly state which instructions can be executed in parallel. Outline its disadvantages, such as time-consuming coding, missed opportunities for parallel processing, and errors due to mistaken indication of parallel processing.

3. Explain how the concept of implicit parallelism solves the problems of explicit parallelism. Discuss the four cases, as described on pages 191-192 in the text, where implicit parallelism is useful in removing complexities. These include working with array operations within loops, performing matrix multiplication, conducting parallel searches in databases, and sorting or merging files.

## Threads and Concurrent Programming

1. Remind students of the concept of threads. Explain how threads are useful in minimizing the overhead incurred in the swapping of a process between main memory and secondary storage during its execution.
2. Point out that each active thread in a process has its own processor registers, program counter, stack, and status, and each active thread shares data area and the resources allocated to its process.
3. Explain how the use of threads can improve performance and interactivity. Use an example of a Web server as mentioned on page 193 in the text.

<b>Teaching Tip</b>	For information on concurrent programming in Java, download the article at: <a href="http://www.informit.com/articles/article.asp?p=167821">www.informit.com/articles/article.asp?p=167821</a>
---------------------	--

## Thread States

1. Outline different states that a thread takes as it moves through the system from READY to RUNNING. Use Figure 6.6 on page 194 in the text to explain different thread states. Describe every possible transition from one state to another. Discuss the importance of synchronization in case of multiple threads.

## Thread Control Block

1. Provide students with an overview of the Thread Control Blocks (TCBs) that contain basic information about a thread, such as its ID, state, and a pointer to the process that created it.
2. Describe briefly the contents of the TCB using Figure 6.7 and the bullet points in the text on page 195 in the text.

## Concurrent Programming Languages

1. Provide a brief introduction to the two programming languages, Ada and Java.

<b>Teaching Tip</b>	Refer to the following Web site to learn more about Ada language: <a href="http://www.adahome.com/rm95">www.adahome.com/rm95</a>
---------------------	---

## Java

1. Provide students with an overview of the Java language, which was the first software platform that promised to allow programmers to code an application once that would run on any computer. Point out that Java uses both a compiler and an interpreter, which makes it easy to distribute Java applications.
2. Discuss several issues that Java solved, such as the high cost of developing software applications for different incompatible computer architectures, the needs of distributed client-server environments, and the issues related to the growth of the Internet and the World Wide Web.
3. Explain the Java platform, which is a software-only platform that runs on top of other hardware-based platforms. Discuss briefly the two components of Java, namely the Java Virtual Machine (Java VM) and the Java Application Programming Interface (Java API).
4. Discuss various features of the Java language, such as that it looks and feels like C++, is object oriented, fits well into distributed client-server applications, offers run-time memory allocation, provides compile-time checking and run-time checking, has sophisticated synchronization capabilities, and supports multithreading at the language level.
5. Outline several reasons why Java technology gained popularity among programmers. Use the bullet points listed on page 197 in the text.

<b>Teaching Tip</b>	Refer to the following Web site to get detailed information about Java technology including popular downloads, technical articles, learning tools, and specifications, etc.: <a href="http://java.sun.com/">http://java.sun.com/</a>
<b>Teaching Tip</b>	Refer to the following Web site to download free Java software: <a href="http://www.java.com/en">www.java.com/en</a>

## **Quick Quiz 2**

1. Which of the following conditions is enforced by using P and V operations on semaphores?
  - a. No preemption
  - b. Resource holding
  - c. Mutual exclusion
  - d. StarvationAnswer: c
  
2. Which of the following truly describes threads? (Choose all that apply.)
  - a. Increases overhead
  - b. Each active thread in a process does not possess its own processor registers, program counter, stack and status
  - c. A smaller unit within a process, which can be scheduled and executed
  - d. Each active thread shares data area and the resources allocated to its processAnswer: c and d
  
3. \_\_\_\_\_ is a type of concurrent programming where the compiler automatically detects which instructions can be performed in parallel.  
Answer: Implicit parallelism

## **Class Discussion Topics**

1. Have students discuss the importance of mutual exclusion and synchronization in the case of process cooperation. What problems may arise when the system lacks these two features? Let students discuss such cases and how each can be solved using semaphores.
  
2. Have students compare threads with processes. What similarities and differences do they notice?

## **Additional Projects**

1. Have students research the type of multiprocessor configuration IBM uses in its recent mainframe computers. Ask students to select and name one of the recent mainframes and compile a list of its key features.
  
2. Have students research how to write an algorithm for an airline reservation system using semaphore variables.

## **Additional Resources**

1. Sun Microsystems:  
[www.sun.com/](http://www.sun.com/)

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

## Key Terms

- **Busy waiting:** a method by which processes, waiting for an event to occur, continuously test to see if the condition has changed and remain in unproductive, resource-consuming wait loops.
- **COBEGIN:** command used with COEND to indicate to a multiprocessing compiler the beginning of a section where instructions can be processed concurrently.
- **COEND:** command used with COBEGIN to indicate to a multiprocessing compiler the end of a section where instructions can be processed concurrently.
- **Concurrent processing:** execution by a single processor of a set of processes in such a way that they appear to be happening at the same time.
- **Concurrent programming:** a programming technique that allows a single processor to simultaneously execute multiple sets of instructions.
- **Critical region:** a part of a program that must complete execution before other processes can have access to the resources being used.
- **Explicit parallelism:** a type of concurrent programming that requires that the programmer explicitly state which instructions can be executed in parallel.
- **Implicit parallelism:** a type of concurrent programming in which the compiler automatically detects which instructions can be performed in parallel.
- **Java:** a cross-platform programming language, developed by Sun Microsystems, that closely resembles C++ and runs on any computer capable of running the Java interpreter.
- **Loosely coupled configuration:** a multiprocessing configuration in which each processor has a copy of the operating system and controls its own resources.
- **Master/slave:** an asymmetric multiprocessing configuration consisting of a single processor system connected to “slave” processors, each of which is managed by the primary “master” processor, which provides the scheduling functions and jobs.
- **Multiprocessing:** when two or more CPUs share the same main memory, most I/O devices, and the same control program routines.
- **Mutex:** a condition that specifies that only one process may update (modify) a shared resource at a time to ensure correct operation and results.

- **Order of operations:** the algebraic convention that dictates the order in which elements of a formula are calculated. Also called precedence of operations or rules of precedence.
- **Parallel processing:** the process of operating two or more CPUs in parallel, with more than one CPU executing instructions simultaneously.
- **Pointer:** an address or other indicator of location.
- **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.
- **Producers and consumers:** a classic problem in which a process produces data that will be consumed, or used, by another process.
- **Readers and writers:** a problem that arises when two types of processes need to access a shared resource such as a file or a database.
- **Semaphore:** a type of shared data item that may contain either binary or nonnegative integer values and is used to provide mutual exclusion.
- **Symmetric configuration:** a multiprocessing configuration in which processor scheduling is decentralized and each processor is of the same type.
- **Test-and-set (TS):** an indivisible machine instruction, which is executed in a single machine cycle to determine whether the processor is available.
- **Thread Control Block (TCB):** a data structure that contains information about the current status and characteristics of a thread.
- **WAIT and SIGNAL:** a modification of the test-and-set synchronization mechanism that's designed to remove busy waiting.