In Chapter 6, we presented the critical-section problem and focused on how race conditions can occur when multiple concurrent processes share data. We went on to examine several tools that address the critical-section problem by preventing race conditions from occurring. These tools ranged from low-level hardware solutions (such as memory barriers and the compare-and-swap operation) to increasingly higher-level tools (from mutex locks to semaphores to monitors). We also discussed various challenges in designing applications that are free from race conditions, including liveness hazards such as deadlocks. In this chapter, we apply the tools presented in Chapter 6 to several classic synchronization problems. We also explore the synchronization mechanisms used by the Linux, UNIX, and Windows operating systems, and we describe API details for both Java and POSIX systems.

Bibliographical Notes


Bibliography

Chapter 7  Synchronization Examples


