A cooperating process is one that can affect or be affected by other processes executing in the system. Cooperating processes can either directly share a logical address space (that is, both code and data) or be allowed to share data only through files or messages. The former case is achieved through the use of threads, discussed in Chapter 4. Concurrent access to shared data may result in data inconsistency, however. In this chapter, we discuss various mechanisms to ensure the orderly execution of cooperating processes that share a logical address space, so that data consistency is maintained.

Bibliographical Notes

The mutual-exclusion problem was first discussed in a classic paper by [Dijkstra (1965)]. Dekker’s algorithm (Exercise 5.8)—the first correct software solution to the two-process mutual-exclusion problem—was developed by the Dutch mathematician T. Dekker. This algorithm also was discussed by [Dijkstra (1965)]. A simpler solution to the two-process mutual-exclusion problem has since been presented by [Peterson (1981)] (Figure 5.2). The semaphore concept was suggested by [Dijkstra (1965)].

The classic process-coordination problems that we have described are paradigms for a large class of concurrency-control problems. The bounded-buffer problem and the dining-philosophers problem were suggested by [Dijkstra (1965)] and [Dijkstra (1971)]. The readers–writers problem was suggested by [Courtois et al. (1971)].

The critical-region concept was suggested by [Hoare (1972)] and by [Brinch-Hansen (1972)]. The monitor concept was developed by [Brinch-Hansen (1973)]. [Hoare (1974)] gave a complete description of the monitor.

Some details of the locking mechanisms used in Solaris were presented in [Mauro and McDougall (2007)]. As noted earlier, the locking mechanisms used by the kernel are implemented for user-level threads as well, so the same types of locks are available inside and outside the kernel. Details of Windows 2000 synchronization can be found in [Solomon and Russinovich (2000)]. [Love (2010)] describes synchronization in the Linux kernel.
Chapter 5  Process Synchronization

Information on Pthreads programming can be found in [Lewis and Berg (1998)] and [Butenhof (1997)]. [Hart (2005)] describes thread synchronization using Windows. [Goetz et al. (2006)] presents a detailed discussion of concurrent programming in Java as well as the java.util.concurrent package. [Breshears (2009)] and [Pacheco (2011)] provide detailed coverage of synchronization issues in relation to parallel programming. [Lu et al. (2008)] provides a study of concurrency bugs in real-world applications.


Bibliography


