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 shared memory or message passing. 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)]. The critical-region concept was suggested by [Hoare (1972)] and by [Brinch-Hansen (1972)]. Dekker’s algorithm (Exercise 6.13)—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 6.3) A thorough discussion of memory barriers and cache memory is presented in [Mckenney (2010)]. [Herlihy and Shavit (2012)] presents details on several issues related to multiprocessor programming, including memory models and compare-and-swap instructions. [Bahra (2013)] examines nonblocking algorithms on modern multicore systems. The semaphore concept was suggested by [Dijkstra (1965)]. The monitor concept was developed by [Brinch-Hansen (1973)]. [Hoare (1974)] gave a complete description of the monitor. [Lu et al. (2008)] provide a study of concurrency bugs in real-world applications. The algorithm proposed in Exercise 6.10 is derived from [Treiber (1986)].

Bibliography

Chapter 6  Synchronization Tools

*Communications of the ACM*, Volume 15, Number 7 (1972), pages 574–578.


