CHAPTER 5

Process Synchronization

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 5.10)—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.

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.

[Adl-Tabatabai et al. (2007)] discuss transactional memory. Details on using OpenMP can be found at `http://openmp.org`. Functional programming using Erlang and Scala is covered in [Armstrong (2007)] and [Odersky et al. ()] respectively.

[Dijkstra (1965)] was one of the first and most influential contributors in the deadlock area. A study of deadlock handling is provided in [Levine (2003)].

Bibliography

[Adl-Tabatabai et al. (2007)] A.-R. Adl-Tabatabai, C. Kozyrakis, and B. Saha, "Unlocking Concurrency", *Queue*, Volume 4, Number 10 (2007), pages 24–33.

[Armstrong (2007)] J. Armstrong, *Programming Erlang Software for a Concurrent World*, The Pragmatic Bookshelf (2007).

[Breshears (2009)] C. Breshears, *The Art of Concurrency*, O'Reilly & Associates (2009).

[Brinch-Hansen (1972)] P. Brinch-Hansen, "Structured Multiprogramming", *Communications of the ACM*, Volume 15, Number 7 (1972), pages 574–578.

[Brinch-Hansen (1973)] P. Brinch-Hansen, *Operating System Principles*, Prentice Hall (1973).

[Butenhof (1997)] D. Butenhof, *Programming with POSIX Threads*, Addison-Wesley (1997).

[Courtois et al. (1971)] P. J. Courtois, F. Heymans, and D. L. Parnas, "Concurrent Control with 'Readers' and 'Writers'", *Communications of the ACM*, Volume 14, Number 10 (1971), pages 667–668.

[Dijkstra (1965)] E. W. Dijkstra, "Cooperating Sequential Processes", Technical report, Technological University, Eindhoven, the Netherlands (1965).

[Dijkstra (1971)] E. W. Dijkstra, "Hierarchical Ordering of Sequential Processes", *Acta Informatica*, Volume 1, Number 2 (1971), pages 115–138.

[Goetz et al. (2006)] B. Goetz, T. Peirls, J. Bloch, J. Bowbeer, D. Holmes, and D. Lea, *Java Concurrency in Practice*, Addison-Wesley (2006).

[Hart (2005)] J. M. Hart, *Windows System Programming*, Third Edition, Addison-Wesley (2005).

[Hoare (1972)] C. A. R. Hoare, "Towards a Theory of Parallel Programming", in **[Hoare and Perrott 1972]** (1972), pages 61–71.

[Hoare (1974)] C. A. R. Hoare, "Monitors: An Operating System Structuring Concept", *Communications of the ACM*, Volume 17, Number 10 (1974), pages 549–557.

[Levine (2003)] G. Levine, "Defining Deadlock", *Operating Systems Review*, Volume 37, Number 1 (2003).

[Lewis and Berg (1998)] B. Lewis and D. Berg, *Multithreaded Programming with Pthreads*, Sun Microsystems Press (1998).

[Love (2010)] R. Love, *Linux Kernel Development,* Third Edition, Developer's Library (2010).

[Lu et al. (2008)] S. Lu, S. Park, E. Seo, and Y. Zhou, "Learning from mistakes: a comprehensive study on real world concurrency bug characteristics", *SIGPLAN Notices*, Volume 43, Number 3 (2008), pages 329–339.

[Mauro and McDougall (2007)] J. Mauro and R. McDougall, *Solaris Internals: Core Kernel Architecture*, Prentice Hall (2007).

[Odersky et al. ()] M. Odersky, V. Cremet, I. Dragos, G. Dubochet, B. Emir, S. Mcdirmid, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, L. Spoon, and M. Zenger.

[Pacheco (2011)] P. S. Pacheco, *An Introduction to Parallel Programming*, Morgan Kaufmann (2011).

[Peterson (1981)] G. L. Peterson, "Myths About the Mutual Exclusion Problem", *Information Processing Letters*, Volume 12, Number 3 (1981).

[Solomon and Russinovich (2000)] D. A. Solomon and M. E. Russinovich, *Inside Microsoft Windows 2000,* Third Edition, Microsoft Press (2000).