Synchronization Examples

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

Details of Windows 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)] present 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. [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. (2006)], respectively. Both [Oaks (2014)] and [Goetz et al. (2006)] contrast traditional synchronization and CAS-based strategies in Java.

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).

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

[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).

[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).

[Oaks (2014)] S. Oaks, *Java Performance - The Definitive Guide*, O'Reilly & Associates (2014).

[Odersky et al. (2006)] 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, "An overview of the Scala programming language", *EPFL* (2006).

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

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

[Treiber (1986)] R. K. Treiber, "Systems Programming: Coping with Parallelism" (1986).