CHAPTER 14

File-System Implementation

As we saw in Chapter 13, the file system provides the mechanism for on-line storage and access to file contents, including data and programs. File systems usually reside permanently on secondary storage, which is designed to hold a large amount of data. This chapter is primarily concerned with issues surrounding file storage and access on the most common secondary-storage media, hard disk drives and non-volatile memory. We explore ways to structure file use, to allocate storage space, to recover freed space, to track the locations of data, and to interface other parts of the operating system to secondary storage. Performance issues are considered throughout the chapter.

A given general purpose operating system provides several file systems. Additionally, many operating systems allow the administrators or users to add file systems. Why so many? File systems vary aspects including features, performance, reliability, and design goals. For example, a temporary file system is used for fast storage and retrieval of non-persistent files, while the default secondary storage file system (such as Linux ext4) sacrifices performance for reliability and features. As we've seen in many other examples throughout this study of operating systems, there are plenty of choices and variations, making thorough coverage a challenge. In this chapter we concentrate on the common denominators.

Bibliographical Notes

The MS-DOS FAT system is explained in [Norton and Wilton (1988)]. The internals of the BSD UNIX system are covered in full in [McKusick and Neville-Neil (2005)]. Details concerning file systems for Linux can be found in [Love (2010)]. The Google file system is described in [Ghemawat et al. (2003)]. FUSE can be found at http://fuse.sourceforge.net.

Log-structured file organizations for enhancing both performance and consistency are discussed in [Rosenblum and Ousterhout (1991)], [Seltzer et al. (1993)], and [Seltzer et al. (1995)]. Algorithms such as balanced trees (and much more) are covered by [Knuth (1998)] and [Cormen et al. (2009)]. [Silvers (2000)] discusses implementing the page cache in the NetBSD operating system. The ZFS source code for space maps can be found at

http://src.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/uts/common/fs/zfs/space_map.c.

The network file system (NFS) is discussed in [Callaghan (2000)]. NFS Version 4 is a standard described at http://www.ietf.org/rfc/rfc3530.txt. [Ousterhout (1991)] discusses the role of distributed state in networked file systems. Log-structured designs for networked file systems are proposed in [Hartman and Ousterhout (1995)] and [Thekkath et al. (1997)]. NFS and the UNIX file system (UFS) are described in [Vahalia (1996)] and [Mauro and McDougall (2007)]. The NTFS file system is explained in [Solomon (1998)]. The Ext3 file system used in Linux is described in [Mauerer (2008)] and the WAFL file system is covered in [Hitz et al. (1995)]. ZFS documentation can be found at http://www.opensolaris.org/os/community/ZFS/docs.

Bibliography

[Callaghan (2000)] B. Callaghan, *NFS Illustrated*, Addison-Wesley (2000).

[Cormen et al. (2009)] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Third Edition, MIT Press (2009).

[Ghemawat et al. (2003)] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System", *Proceedings of the ACM Symposium on Operating Systems Principles* (2003).

[Hartman and Ousterhout (1995)] J. H. Hartman and J. K. Ousterhout, "The Zebra Striped Network File System", *ACM Transactions on Computer Systems*, Volume 13, Number 3 (1995), pages 274–310.

[Hitz et al. (1995)] D. Hitz, J. Lau, and M. Malcolm, "File System Design for an NFS File Server Appliance", Technical report, NetApp (1995).

[Knuth (1998)] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Second Edition, Addison-Wesley (1998).

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

[Mauerer (2008)] W. Mauerer, *Professional Linux Kernel Architecture*, John Wiley and Sons (2008).

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

[McKusick and Neville-Neil (2005)] M. K. McKusick and G. V. Neville-Neil, *The Design and Implementation of the FreeBSD UNIX Operating System*, Addison Wesley (2005).

[Norton and Wilton (1988)] P. Norton and R. Wilton, *The New Peter Norton Programmer's Guide to the IBM PC & PS/2*, Microsoft Press (1988).

[Ousterhout (1991)] J. Ousterhout. "The Role of Distributed State". *In CMU Computer Science: a 25th Anniversary Commemorative*, R. F. Rashid, Ed., Addison-Wesley (1991).

[Rosenblum and Ousterhout (1991)] M. Rosenblum and J. K. Ousterhout, "The Design and Implementation of a Log-Structured File System", *Proceedings of the ACM Symposium on Operating Systems Principles* (1991), pages 1–15.

[Seltzer et al. (1993)] M. I. Seltzer, K. Bostic, M. K. McKusick, and C. Staelin, "An Implementation of a Log-Structured File System for UNIX", *USENIX Winter* (1993), pages 307–326.

[Seltzer et al. (1995)] M. I. Seltzer, K. A. Smith, H. Balakrishnan, J. Chang, S. McMains, and V. N. Padmanabhan, "File System Logging Versus Clustering: A Performance Comparison", *USENIX Winter* (1995), pages 249–264.

[Silvers (2000)] C. Silvers, "UBC: An Efficient Unified I/O and Memory Caching Subsystem for NetBSD", *USENIX Annual Technical Conference—FREENIX Track* (2000).

[Solomon (1998)] D. A. Solomon, *Inside Windows NT,* Second Edition, Microsoft Press (1998).

[Thekkath et al. (1997)] C. A. Thekkath, T. Mann, and E. K. Lee, "Frangipani: A Scalable Distributed File System", *Symposium on Operating Systems Principles* (1997), pages 224–237.

[Vahalia (1996)] U. Vahalia, *Unix Internals: The New Frontiers*, Prentice Hall (1996).