Preface

Operating systems are an essential part of any computer system. Similarly, a course on operating systems is an essential part of any computer-science education. This field is undergoing rapid change, as computers are now prevalent in virtually every application, from games for children through the most sophisticated planning tools for governments and multinational firms. Yet the fundamental concepts remain fairly clear, and it is on these that we base this book.

We wrote this book as a text for an introductory course in operating systems at the junior or senior undergraduate level or at the first-year graduate level. We hope that practitioners will also find it useful. It provides a clear description of the *concepts* that underlie operating systems. As prerequisites, we assume that the reader is familiar with basic data structures, computer organization, and a high-level language, such as C. The hardware topics required for an understanding of operating systems are included in Chapter 1. For code examples, we use predominantly C, with some Java, but the reader can still understand the algorithms without a thorough knowledge of these languages.

Concepts are presented using intuitive descriptions. Important theoretical results are covered, but formal proofs are omitted. The bibliographical notes contain pointers to research papers in which results were first presented and proved, as well as references to material for further reading. In place of proofs, figures and examples are used to suggest why we should expect the result in question to be true.

The fundamental concepts and algorithms covered in the book are often based on those used in existing commercial operating systems. Our aim is to present these concepts and algorithms in a general setting that is not tied to one particular operating system. We present a large number of examples that pertain to the most popular and the most innovative operating systems, including Sun Microsystems' Solaris; Linux; Mach; Microsoft MS-DOS, Windows NT, Windows 2000, and Windows XP; DEC VMS and TOPS-20; IBM OS/2; and Apple Mac OS X.

In this text, when we refer to Windows XP as an example operating system, we are implying both Windows XP and Windows 2000. If a feature exists in Windows XP that is not available in Windows 2000, we will state this explicitly.

If a feature exists in Windows 2000 but not in Windows XP, then we will refer specifically to Windows 2000.

Organization of This Book

The organization of this text reflects our many years of teaching operating systems courses. Consideration was also given to the feedback provided by the reviewers of the text, as well as comments submitted by readers of earlier editions. In addition, the content of the text corresponds to the suggestions from *Computing Curricula 2001* for teaching operating systems, published by the Joint Task Force of the IEEE Computing Society and the Association for Computing Machinery (ACM).

On the supporting web page for this text, we provide several sample syllabi that suggest various approaches for using the text in both introductory and advanced operating systems courses. As a general rule, we encourage readers to progress sequentially through the chapters, as this strategy provides the most thorough study of operating systems. However, by using the sample syllabi, a reader can select a different ordering of chapters (or subsections of chapters).

Content of This Book

The text is organized in eight major parts:

- **Overview**. Chapters 1 and 2 explain what operating systems *are*, what they *do*, and how they are *designed* and *constructed*. They discuss what the common features of an operating system are, what an operating system does for the user, and what it does for the computer-system operator. The presentation is motivational and explanatory in nature. We have avoided a discussion of how things are done internally in these chapters. Therefore, they are suitable for individual readers or for students in lower-level classes who want to learn what an operating system is without getting into the details of the internal algorithms.
- **Process management**. Chapters 3 through 7 describe the process concept and concurrency as the heart of modern operating systems. A *process* is the unit of work in a system. Such a system consists of a collection of *concurrently* executing processes, some of which are operating-system processes (those that execute system code) and the rest of which are user processes (those that execute user code). These chapters cover methods for process scheduling, interprocess communication, process synchronization, and deadlock handling. Also included under this topic is a discussion of threads.
- **Memory management**. Chapters 8 and 9 deal with main memory management during the execution of a process. To improve both the utilization of the CPU and the speed of its response to its users, the computer must keep several processes in memory. There are many different memory-management schemes, reflecting various approaches to memory management, and the effectiveness of a particular algorithm depends on the situation.

- **Storage management**. Chapters 10 through 13 describe how the file system, mass storage, and I/O are handled in a modern computer system. The file system provides the mechanism for on-line storage of and access to both data and programs residing on the disks. These chapters describe the classic internal algorithms and structures of storage management. They provide a firm practical understanding of the algorithms used—the properties, advantages, and disadvantages. Since the I/O devices that attach to a computer vary widely, the operating system needs to provide a wide range of functionality to applications to allow them to control all aspects of the devices. We discuss system I/O in depth, including I/O system design, interfaces, and internal system structures and functions. In many ways, I/O devices are also the slowest major components of the computer. Because they are a performance bottleneck, performance issues are examined. Matters related to secondary and tertiary storage are explained as well.
- **Protection and security**. Chapters 14 and 15 discuss the processes in an operating system that must be protected from one another's activities. For the purposes of protection and security, we use mechanisms that ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory, CPU, and other resources. Protection is a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide a means of specifying the controls to be imposed, as well as a means of enforcement. Security protects the information stored in the system (both data and code), as well as the physical resources of the computer system, from unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.
- **Distributed systems**. Chapters 16 through 18 deal with a collection of processors that do not share memory or a clock—a *distributed system*. By providing the user with access to the various resources that it maintains, a distributed system can improve computation speed and data availability and reliability. Such a system also provides the user with a distributed file system, which is a file-service system whose users, servers, and storage devices are dispersed among the sites of a distributed system. A distributed system must provide various mechanisms for process synchronization and communication and for dealing with the deadlock problem and a variety of failures that are not encountered in a centralized system.
- **Special-purpose systems**. Chapters 19 and 20 deal with systems used for specific purposes, including real-time systems and multimedia systems. These systems have specific requirements that differ from those of the general-purpose systems that are the focus of the remainder of the text. Real-time systems may require not only that computed results be "correct" but also that the results be produced within a specified deadline period. Multimedia systems require quality-of-service guarantees ensuring that the multimedia data are delivered to clients within a specific time frame.
- **Case studies**. Chapters 21 through 23 in the book, and Appendices A through C on the website, integrate the concepts described in this book by describing real operating systems. These systems include Linux, Windows

XP, FreeBSD, Mach, and Windows 2000. We chose Linux and FreeBSD because UNIX—at one time—was almost small enough to understand yet was not a "toy" operating system. Most of its internal algorithms were selected for *simplicity*, rather than for speed or sophistication. Both Linux and FreeBSD are readily available to computer-science departments, so many students have access to these systems. We chose Windows XP and Windows 2000 because they provide an opportunity for us to study a modern operating system with a design and implementation drastically different from those of UNIX. Chapter 23 briefly describes a few other influential operating systems.

Operating-System Environments

This book uses examples of many real-world operating systems to illustrate fundamental operating-system concepts. However, particular attention is paid to the Microsoft family of operating systems (including Windows NT, Windows 2000, and Windows XP) and various versions of UNIX (including Solaris, BSD, and Mac OS X). We also provide a significant amount of coverage of the Linux operating system reflecting the most recent version of the kernel—Version 2.6—at the time this book was written.

The text also provides several example programs written in C and Java. These programs are intended to run in the following programming environments:

- **Windows systems**. The primary programming environment for Windows systems is the Win32 API (**a**pplication **p**rogramming **i**nterface), which provides a comprehensive set of functions for managing processes, threads, memory, and peripheral devices. We provide several C programs illustrating the use of the Win32 API. Example programs were tested on systems running Windows 2000 and Windows XP.
- **POSIX**. POSIX (which stands for *Portable Operating System Interface*) represents a set of standards implemented primarily for UNIX-based operating systems. Although Windows XP and Windows 2000 systems can also run certain POSIX programs, our coverage of POSIX focuses primarily on UNIX and Linux systems. POSIX-compliant systems must implement the POSIX core standard (POSIX.1)—Linux, Solaris, and Mac OS X are examples of POSIX-compliant systems. POSIX also defines several extensions to the standards, including real-time extensions (POSIX1.b) and an extension for a threads library (POSIX1.c, better known as Pthreads). We provide several programming examples written in C illustrating the POSIX base API, as well as Pthreads and the extensions for real-time programming. These example programs were tested on Debian Linux 2.4 and 2.6 systems, Mac OS X, and Solaris 9 using the `gcc` 3.3 compiler.
- **Java**. Java is a widely used programming language with a rich API and built-in language support for thread creation and management. Java programs run on any operating system supporting a Java virtual machine (or JVM). We illustrate various operating system and networking concepts with several Java programs tested using the Java 1.4 JVM.

We have chosen these three programming environments because it is our opinion that they best represent the two most popular models of operating systems: Windows and UNIX/Linux, along with the widely used Java environment. Most programming examples are written in C, and we expect readers to be comfortable with this language; readers familiar with both the C and Java languages should easily understand most programs provided in this text.

In some instances—such as thread creation—we illustrate a specific concept using all three programming environments, allowing the reader to contrast the three different libraries as they address the same task. In other situations, we may use just one of the APIs to demonstrate a concept. For example, we illustrate shared memory using just the POSIX API; socket programming in TCP/IP is highlighted using the Java API.

The Seventh Edition

As we wrote this seventh edition of *Operating System Concepts*, we were guided by the many comments and suggestions we received from readers of our previous editions, as well as by our own observations about the rapidly changing fields of operating systems and networking. We have rewritten the material in most of the chapters by bringing older material up to date and removing material that was no longer of interest or relevance.

We have made substantive revisions and organizational changes in many of the chapters. Most importantly, we have completely reorganized the overview material in Chapters 1 and 2 and have added two new chapters on special-purpose systems (real-time embedded systems and multimedia systems). Because protection and security have become more prevalent in operating systems, we now cover these topics earlier in the text. Moreover, we have substantially updated and expanded the coverage of security.

Below, we provide a brief outline of the major changes to the various chapters:

- **Chapter 1, Introduction,** has been totally revised. In previous editions, the chapter gave a historical view of the development of operating systems. The new chapter provides a grand tour of the major operating-system components, along with basic coverage of computer-system organization.
- **Chapter 2, Operating-System Structures,** is a revised version of old Chapter 3, with many additions, including enhanced discussions of system calls and operating-system structure. It also provides significantly updated coverage of virtual machines.
- **Chapter 3, Processes,** is the old Chapter 4. It includes new coverage of how processes are represented in Linux and illustrates process creation using both the POSIX and Win32 APIs. Coverage of shared memory is enhanced with a program illustrating the shared-memory API available for POSIX systems.
- **Chapter 4, Threads,** is the old Chapter 5. The chapter presents an enhanced discussion of thread libraries, including the POSIX, Win32 API, and Java thread libraries. It also provides updated coverage of threading in Linux.

- **Chapter 5, CPU Scheduling,** is the old Chapter 6. The chapter offers a significantly updated discussion of scheduling issues for multiprocessor systems, including processor affinity and load-balancing algorithms. It also features a new section on thread scheduling, including Pthreads, and updated coverage of table-driven scheduling in Solaris. The section on Linux scheduling has been revised to cover the scheduler used in the 2.6 kernel.
- **Chapter 6, Process Synchronization,** is the old Chapter 7. We have removed the coverage of two-process solutions and now discuss only Peterson's solution, as the two-process algorithms are not guaranteed to work on modern processors. The chapter also includes new sections on synchronization in the Linux kernel and in the Pthreads API.
- **Chapter 7, Deadlocks,** is the old Chapter 8. New coverage includes a program example illustrating deadlock in a multithreaded Pthread program.
- **Chapter 8, Main Memory,** is the old Chapter 9. The chapter no longer covers overlays. In addition, the coverage of segmentation has seen significant modification, including an enhanced discussion of segmentation in Pentium systems and a discussion of how Linux is designed for such segmented systems.
- **Chapter 9, Virtual Memory,** is the old Chapter 10. The chapter features expanded coverage of motivating virtual memory as well as coverage of memory-mapped files, including a programming example illustrating shared memory (via memory-mapped files) using the Win32 API. The details of memory management hardware have been modernized. A new section on allocating memory within the kernel discusses the buddy algorithm and the slab allocator.
- **Chapter 10, File-System Interface,** is the old Chapter 11. It has been updated and an example of Windows XP ACLs has been added.
- **Chapter 11, File-System Implementation,** is the old Chapter 12. Additions include a full description of the WAFL file system and inclusion of Sun's ZFS file system.
- **Chapter 12, Mass-Storage Structure,** is the old Chapter 14. New is the coverage of modern storage arrays, including new RAID technology and features such as thin provisioning.
- **Chapter 13, I/O Systems,** is the old Chapter 13 updated with coverage of new material.
- **Chapter 14, Protection,** is the old Chapter 18 updated with coverage of the principle of least privilege.
- **Chapter 15, Security,** is the old Chapter 19. The chapter has undergone a major overhaul, with all sections updated. A full example of a buffer-overflow exploit is included, and coverage of threats, encryption, and security tools has been expanded.
- **Chapters 16 through 18** are the old Chapters 15 through 17, updated with coverage of new material.

- **Chapter 19, Real-Time Systems,** is a new chapter focusing on real-time and embedded computing systems, which have requirements different from those of many traditional systems. The chapter provides an overview of real-time computer systems and describes how operating systems must be constructed to meet the stringent timing deadlines of these systems.
- **Chapter 20, Multimedia Systems,** is a new chapter detailing developments in the relatively new area of multimedia systems. Multimedia data differ from conventional data in that multimedia data—such as frames of video—must be delivered (streamed) according to certain time restrictions. The chapter explores how these requirements affect the design of operating systems.
- **Chapter 21, The Linux System,** is the old Chapter 20, updated to reflect changes in the 2.6 kernel—the most recent kernel at the time this text was written.
- **Chapter 22, XP,** has been updated.
- **Chapter 22, Influential Operating Systems,** has been updated.

The old Chapter 21 (Windows 2000) has been turned into **Appendix C**. As in the previous edition, the appendices are provided online.

Programming Exercises and Projects

To emphasize the concepts presented in the text, we have added several programming exercises and projects that use the POSIX and Win32 APIsas well as Java. We have added over 15 new programming exercises that emphasize processes, threads, shared memory, process synchronization, and networking. In addition, we have added several programming projects which are more involved than standard programming exercises. These projects include adding a system call to the Linux kernel, creating a UNIX shell using the `fork()` system call, a multithreaded matrix application, and the producer-consumer problem using shared memory.

Teaching Supplements and Web Page

The web page for the book contains such material as a set of slides to accompany the book, model course syllabi, all C and Java source code, and up-to-date errata. The web page also contains the book's three case-study appendices and the Distributed Communication appendix. The URL is:

http://www.os-book.com

New to this edition is a supplement called Practice Exercises, which consists of exercises not found in the text. The Practice Exercises and their associated solutions are publicly available on the Web page of the book. Students are encouraged to solve the practice exercises on their own, and later use the solutions on the Web page to check their own solutions.

To obtain restricted supplements, such as the solution guide to the exercises in the text, contact your local John Wiley & Sons sales representative. Note that these supplements are avaialble only to faculty who use this text. You can find your representative at the "Find a Rep?" web page: http://www.jsw-edcv.wiley.com/college/findarep.

Mailing List

We have switched to the mailman system for communication among the users of *Operating System Concepts*. If you wish to use this facility, please visit the following URL and follow the instructions there to subscribe:

http://mailman.cs.yale.edu/mailman/listinfo/os-book-list

The mailman mailing-list system provides many benefits, such as an archive of postings, as well as several subscription options, including digest and Web only. To send messages to the list, send e-mail to:

os-book-list@cs.yale.edu

Depending on the message, we will either reply to you personally or forward the message to everyone on the mailing list. The list is moderated, so you will receive no inappropriate mail.

Students who are using this book as a text for class should not use the list to ask for answers to the exercises. They will not be provided.

Suggestions

We have attempted to clean up every error in this new edition, but—as happens with operating systems—a few obscure bugs may remain. We would appreciate hearing from you about any textual errors or omissions that you identify.

If you would like to suggest improvements or to contribute exercises, we would also be glad to hear from you. Please send correspondence to os-book@cs.yale.edu.

Acknowledgments

This book is derived from the previous editions, the first three of which were coauthored by James Peterson. Others who helped us with previous editions include Hamid Arabnia, Rida Bazzi, Randy Bentson, David Black, Joseph Boykin, Jeff Brumfield, Gael Buckley, Roy Campbell, P. C. Capon, John Carpenter, Gil Carrick, Thomas Casavant, Ajoy Kumar Datta, Joe Deck, Sudarshan K. Dhall, Thomas Doeppner, Caleb Drake, M. Racsit Eskicioğlu, Hans Flack, Robert Fowler, G. Scott Graham, Richard Guy, Max Hailperin, Rebecca Hartman, Wayne Hathaway, Christopher Haynes, Bruce Hillyer, Mark Holliday, Ahmed Kamel, Richard Kieburtz, Carol Kroll, Morty Kwestel, Thomas LeBlanc, John Leggett, Jerrold Leichter, Ted Leung, Gary Lippman, Carolyn Miller,

Michael Molloy, Yoichi Muraoka, Jim M. Ng, Banu Özden, Ed Posnak, Boris Putanec, Charles Qualline, John Quarterman, Mike Reiter, Gustavo Rodriguez-Rivera, Carolyn J. C. Schauble, Thomas P. Skinner, Yannis Smaragdakis, Jesse St. Laurent, John Stankovic, Adam Stauffer, Steven Stepanek, Hal Stern, Louis Stevens, Pete Thomas, David Umbaugh, Steve Vinoski, Tommy Wagner, Larry L. Wear, John Werth, James M. Westall, J. S. Weston, and Yang Xiang

Parts of Chapter 12 were derived from a paper by Hillyer and Silberschatz [1996]. Parts of Chapter 17 were derived from a paper by Levy and Silberschatz [1990]. Chapter 21 was derived from an unpublished manuscript by Stephen Tweedie. Chapter 22 was derived from an unpublished manuscript by Dave Probert, Cliff Martin, and Avi Silberschatz. Appendix C was derived from an unpublished manuscript by Cliff Martin. Cliff Martin also helped with updating the UNIX appendix to cover FreeBSD. Mike Shapiro, Bryan Cantrill, and Jim Mauro answered several Solaris-related questions. Josh Dees and Rob Reynolds contributed coverage of Microsoft's .NET. The project for designing and enhancing the UNIX shell interface was contributed by John Trono of St. Michael's College in Winooski, Vermont.

This edition has many new exercises and accompanying solutions, which were supplied by Arvind Krishnamurthy.

We thank the following people who reviewed this version of the book: Bart Childs, Don Heller, Dean Hougen Michael Huangs, Morty Kewstel, Euripides Montagne, and John Sterling.

Our Acquisitions Editors, Bill Zobrist and Paul Crockett, provided expert guidance as we prepared this edition. They were assisted by Simon Durkin, who managed many details of this project smoothly. The Senior Production Editor was Ken Santor. The cover illustrator was Susan Cyr, and the cover designer was Madelyn Lesure. Beverly Peavler copy-edited the manuscript. The freelance proofreader was Katrina Avery; the freelance indexer was Rosemary Simpson. Marilyn Turnamian helped generate figures and presentation slides.

Finally, we would like to add some personal notes. Avi is starting a new chapter in his life, returning to academia and partnering with Valerie. This combination has given him the peace of mind to focus on the writing of this text. Pete would like to thank his family, friends, and coworkers for their support and understanding during the project. Greg would like to acknowledge the continued interest and support from his family. However, he would like to single out his friend Peter Ormsby who—no matter how busy his life seems to be—always first asks, "How's the writing coming along?"

Abraham Silberschatz, New Haven, CT, 2004
Peter Baer Galvin, Burlington, MA, 2004
Greg Gagne, Salt Lake City, UT, 2004