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 arena of day-to-day life—from embedded devices in automobiles 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 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 or Java. The hardware topics required to understand of operating systems are covered in Chapter 1. In that chapter, we also include an overview of the fundamental data structures that are prevalent in most operating systems. For code examples, we use predominantly C, as well as a significant amount of 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 largely omitted. The bibliographical notes at the end of each chapter contain pointers to research papers in which results were first presented and proved, as well as references to recent 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 both open-source and 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. However, we present a large number of examples that pertain to the most popular and the most innovative operating systems, including Linux, Microsoft Windows, Apple macOS (the original name, OS X, was changed in 2016 to match the naming scheme of other Apple products), and Solaris. We also include examples of both Android and iOS, currently the two dominant mobile operating systems.

The organization of the text reflects our many years of teaching courses on operating systems. Consideration was also given to the feedback provided by the reviewers of the text, along with the many comments and suggestions we received from readers of our previous editions and from our current and former students. This Tenth Edition also reflects most of the curriculum guidelines in the operating-systems area in *Computer Science Curricula 2013*, the most recent curriculum guidelines in undergraduate degree programs in computer science published by the IEEE Computing Society and the Association for Computing Machinery (ACM).

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. These chapters discuss what the common features of an operating system are and what an operating system does for the user. We include coverage of both traditional PC and server operating systems and operating systems for mobile devices. The presentation is motivational and explanatory in nature. We have avoided a discussing 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 5 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 executing operating-system code and others executing user code. These chapters cover methods for process scheduling and interprocess communication. Also included is a detailed discussion of threads, as well as an examination of issues related to multicore systems and parallel programming.

- **Process synchronization**. Chapters 6 through 8 cover methods for process synchronization and deadlock handling. Because we have increased the coverage of process synchronization, we have divided the former Chapter 5 (Process Synchronization) into two separate chapters: Chapter 6, Synchronization Tools, and Chapter 7, Synchronization Examples.

- **Memory management**. Chapters 9 and 10 deal with the management of main memory 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 memory-management management, and the effectiveness of a particular algorithm depends on the situation.

- **Storage management**. Chapters 11 and 12 describe how mass storage and I/O are handled in a modern computer system. The I/O devices that attach to a computer vary widely, and the operating system needs to provide a wide range of functionality to applications to let them to control all aspects of these 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 the slowest major components of the computer. Because they represent a performance bottleneck, we also examine performance issues associated with I/O devices.

- **File systems**. Chapters 13 through 15 discuss how file systems are handled in a modern computer system. File systems provide the mechanism for online storage of and access to both data and programs. We describe the classic internal algorithms and structures of storage management and provide a firm practical understanding of the algorithms used—their properties, advantages, and disadvantages.
- **Security and protection**. Chapters 16 and 17 discuss the mechanisms necessary for the security and protection of computer systems. The processes in an operating system must be protected from one another's activities. To provide such protection, we must ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory, CPU, and other system resources. Protection is a mechanism for controlling the access of programs, processes, or users to computer-system resources. This mechanism must provide a means of specifying the controls to be imposed, as well as a means of enforcement. Security protects the integrity of the information stored in the system (both data and code), as well as the physical resources of the system, from unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.
- **Advanced topics**. Chapters 18 and 19 discuss virtual machines and networks/distributed systems. Chapter 18 provides an overview of virtual machines and their relationship to contemporary operating systems. Included is a general description of the hardware and software techniques that make virtualization possible. Chapter 19 provides an overview of computer networks and distributed systems, with a focus on the Internet and TCP/IP.

Programming Environments

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

- **POSIX**. POSIX (which stands for *Portable Operating System Interface*) is a set of standards implemented primarily for UNIX-based operating systems. Although Windows systems can also run certain POSIX programs, our coverage of POSIX focuses on Linux and UNIX systems. POSIX-compliant systems must implement the POSIX core standard (POSIX.1); Linux and macOS are examples of POSIX-compliant systems. POSIX also defines several extensions to the standards, including real-time extensions (POSIX.1b) and an extension for a threads library (POSIX.1c, 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 Linux 4.4 and macOS 10.11 systems using the gcc compiler.

- **Java**. Java is a widely used programming language with a rich API and built-in language support for concurrent and parallel programming. Java programs run on any operating system supporting a Java virtual machine (or JVM). We illustrate various operating-system and networking concepts with Java programs tested using Version 1.8 of the Java Development Kit (JDK).
- **Windows systems**. The primary programming environment for Windows systems is the Windows API, which provides a comprehensive set of functions for managing processes, threads, memory, and peripheral devices. We supply a modest number of C programs illustrating the use of this API. Programs were tested on a system running Windows 10.

We have chosen these three programming environments because we believe that they best represent the two most popular operating-system models—Linux/UNIX and Windows—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 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.

Linux Virtual Machine

To help students gain a better understanding of the Linux system, we provide a Linux virtual machine running the Ubuntu distribution with this text. The virtual machine, which is available for download from the text website (http://www.os-book-global.com), also provides development environments including the gcc and Java compilers. Most of the programming assignments in the book can be completed using this virtual machine; the exception is assignments that require the Windows API. The virtual machine can be installed and run on any host operating system that can run the VirtualBox virtualization software, which currently includes Windows 10 Linux, and macOS.

The Tenth Edition

As we wrote this Tenth Edition of *Operating System Concepts*, we were guided by the sustained growth in four fundamental areas that affect operating systems:

1. Mobile operating systems
2. Multicore systems
3. Virtualization
4. Nonvolatile memory secondary storage

To emphasize these topics, we have integrated relevant coverage throughout this new edition. For example, we have greatly increased our coverage of the Android and iOS mobile operating systems, as well as of the ARMv8 architecture that dominates mobile devices. We have also increased our coverage of multicore systems, including APIs that support concurrency and parallelism. Nonvolatile memory devices like SSDs are now treated as the equals of hard-disk drives in the chapters that discuss I/O, mass storage, and file systems.

Several of our readers have expressed support for an increase in Java coverage, and we have provided additional Java examples throughout this edition.

Additionally, we have rewritten material in almost every chapter by bringing older material up to date and removing material that is no longer interesting or relevant. We have reordered many chapters and have, in some instances, moved sections from one chapter to another. We have also greatly revised the artwork, creating several new figures as well as modifying many existing figures.

Major Changes

The Tenth Edition update encompasses much more material than previous updates, in terms of both content and new supporting material. Here, outline the major content changes in each chapter:

- **Chapter 1: Introduction** includes updated coverage of multicore systems, as well as new coverage of NUMA systems and Hadoop clusters. Old material has been updated, and new motivation has been added for studying of operating systems.
- **Chapter 2: Operating-System Structures** provides a significantly revised discussion of the design and implementation of operating systems. We have updated our treatment of Android and iOS and have revised our coverage of the system boot process with a focus on GRUB for Linux systems. New coverage of the Windows subsystem for Linux is included as well. We have added new sections on linkers and loaders, and we now discuss why applications are often operating-system specific. Finally, we have added a discussion of the BCC debugging toolset.
- **Chapter 3: Processes** simplifies the discussion of scheduling so that it now includes only CPU scheduling issues. New coverage describes the memory layout of a C program, the Android process hierarchy, Mach message passing, and Android RPCs. We have also replaced coverage of the traditional UNIX/Linux `init` process with coverage of `systemd`.
- **Chapter 4: Threads and Concurrency** (previously **Threads**) increases the coverage of support for concurrent and parallel programming at the API and library level. We have revised the section on Java threads so that it now includes Futures and have updated the coverage of Apple's Grand Central Dispatch so that it now includes Swift. New sections discuss fork-join parallelism using the fork-join framework in Java, as well as Intel thread building blocks.

- **Chapter 5: CPU Scheduling** (previously Chapter 6) revises the coverage of multilevel queue and multicore processing scheduling. We have integrated coverage of NUMA-aware scheduling issues throughout, including how this scheduling affects load balancing. We also discuss related modifications to the Linux CFS scheduler. New coverage combines discussions of round-robin and priority scheduling, heterogeneous multiprocessing, and Windows 10 scheduling.
- **Chapter 6: Synchronization Tools** (previously part of Chapter 5, **Process Synchronization**) focuses on various tools for synchronizing processes. Significant new coverage discusses architectural issues such as instruction reordering and delayed writes to buffers. The chapter also introduces lock-free algorithms using compare-and-swap (CAS) instructions. No specific APIs are presented; rather, the chapter provides an introduction to race conditions and general tools that can be used to prevent data races. Details include new coverage of memory models, memory barriers, and liveness issues.
- **Chapter 7: Synchronization Examples** (previously part of Chapter 5, **Process Synchronization**) introduces classical synchronization problems and discusses specific API support for designing solutions that solve these problems. The chapter includes new coverage of POSIX named and unnamed semaphores, as well as condition variables. A new section on Java synchronization is included as well.
- **Chapter 8: Deadlocks** (previously Chapter 7) contains minor updates, including a new section on livelock and a discussion of deadlock as an example of a liveness hazard. The chapter includes new coverage of the Linux `lockdep` and the BCC `deadlock_detector` tools, as well as coverage of Java deadlock detection using thread dumps.
- **Chapter 9: Main Memory** (previously Chapter 8) includes several revisions that bring the chapter up to date on memory management on modern computer systems. We have added new coverage of the ARMv8 64-bit architecture, updated the coverage of dynamic link libraries, and changed swapping coverage so that it now focuses on swapping pages rather than processes. We have also eliminated coverage of segmentation.
- **Chapter 10: Virtual Memory** (previously Chapter 9) contains several revisions, including updated coverage of memory allocation on NUMA systems and global allocation using a free-frame list. New coverage includes compressed memory, major/minor page faults, and memory management in Linux and Windows 10.
- **Chapter 11: Mass-Storage Structure** (previously Chapter 10) adds coverage of nonvolatile memory devices, such as flash and solid-state disks. Hard-drive scheduling is simplified to show only currently used algorithms. Also included are a new section on cloud storage, updated RAID coverage, and a new discussion of object storage.
- **Chapter 12, I/O** (previously Chapter 13) updates the coverage of technologies and performance numbers, expands the coverage of synchronous/asynchronous and blocking/nonblocking I/O, and adds a

section on vectored I/O. It also expands coverage of power management for mobile operating systems.

- **Chapter 13: File-System Interface** (previously Chapter 11) has been updated with information about current technologies. In particular, the coverage of directory structures has been improved, and the coverage of protection has been updated. The memory-mapped files section has been expanded, and a Windows API example has been added to the discussion of shared memory. The topics have been reordered in Chapters 13 and 14.
- **Chapter 14: File-System Implementation** (previously Chapter 12) has been updated with coverage of current technologies. The chapter now includes discussions of TRIM and the Apple File System. In addition, the discussion of performance has been updated, and the coverage of journaling has been expanded.
- **Chapter 15: File System Internals** is new and contains updated information from the previous Chapters 11 and 12.
- **Chapter 16: Security** (previously Chapter 15) now precedes the protection chapter. It includes revised and updated terms for current security threats and solutions, including ransomware and remote access tools. The principle of least privilege is emphasized. Coverage of code-injection vulnerabilities and attacks has been revised and now includes code samples. Discussion of encryption technologies has been updated to focus on the technologies currently used. Coverage of authentication (by passwords and other methods) has been updated and expanded with helpful hints. Additions include a discussion of address-space layout randomization and a new summary of security defenses. The Windows 7 example has been updated to Windows 10.
- **Chapter 17: Protection** (previously Chapter 14) contains major changes. The discussion of protection rings and layers has been updated and now refers to the Bell–LaPadula model and explores the ARM model of TrustZones and Secure Monitor Calls. Coverage of the need-to-know principle has been expanded, as has coverage of mandatory access control. Subsections on Linux capabilities, Darwin entitlements, security integrity protection, system-call filtering, sandboxing, and code signing have been added. Coverage of run-time-based enforcement in Java has also been added, including the stack inspection technique.
- **Chapter 18: Virtual Machines** (previously Chapter 16) includes added details about hardware assistance technologies. Also expanded is the topic of application containment, now including containers, zones, docker, and Kubernetes. A new section discusses ongoing virtualization research, including unikernels, library operating systems, partitioning hypervisors, and separation hypervisors.
- **Chapter 19, Networks and Distributed Systems** (previously Chapter 17) has been substantially updated and now combines coverage of computer networks and distributed systems. The material has been revised to bring it up to date on contemporary computer networks and distributed systems. The TCP/IP model receives added emphasis, and a discussion of

cloud storage has been added. The section on network topologies has been removed. Coverage of name resolution has been expanded and a Java example added. The chapter also includes new coverage of distributed file systems, including MapReduce on top of Google file system, Hadoop, GPFS, and Lustre.

Supporting Website

When you visit the website supporting this text at http://www.os-book-global.com, you can download the following resources:

- Linux virtual machine
- C and Java source code
- The complete set of figures and illustrations
- FreeBSD, Mach, and Windows 7 case studies
- Errata
- Bibliography

Notes to Instructors

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

In this edition, we have added many new written exercises and programming problems and projects. Most of the new programming assignments involve processes, threads, process scheduling, process synchronization, and memory management. Some involve adding kernel modules to the Linux system, which requires using either the Linux virtual machine that accompanies this text or another suitable Linux distribution.

Solutions to written exercises and programming assignments are available to instructors who have adopted this text for their operating-system class. To obtain these restricted supplements, contact your local John Wiley & Sons sales representative. You can find your Wiley representative by going to http://www.wiley.com/college and clicking "Who's my rep?"

Notes to Students

We encourage you to take advantage of the practice exercises that appear at the end of each chapter. We also encourage you to read through the study guide, which was prepared by one of our students. Finally, for students who are unfamiliar with UNIX and Linux systems, we recommend that you download and install the Linux virtual machine that we include on the supporting website. Not only will this provide you with a new computing experience, but the open-source nature of Linux will allow you easily to examine the inner details of this popular operating system. We wish you the very best of luck in your study of operating systems!

Contacting Us

We have endeavored to eliminate typos, bugs, and the like from the text. But, as in new releases of software, bugs almost surely remain. An up-to-date errata list is accessible from the book's website. We would be grateful if you would notify us of any errors or omissions in the book that are not on the current list of errata.

We would be glad to receive suggestions on improvements to the book. We also welcome any contributions to the book website that could be of use to other readers, such as programming exercises, project suggestions, on-line labs and tutorials, and teaching tips. E-mail should be addressed to os-book-authors@cs.yale.edu.

Acknowledgments

Many people have helped us with this Tenth Edition, as well as with the previous nine editions from which it is derived.

Tenth Edition

- Rik Farrow provided expert advice as a technical editor.
- Jonathan Levin helped out with coverage of mobile systems, protection, and security.
- Sarah Diesburg revised Chapter 19: **Networks and Distributed Systems**.
- Brendan Gregg provided guidance on the BCC toolset.
- Richard Stallman (RMS) supplied feedback on the description of free and open-source software.
- Michael Shapiro helped with storage and I/O technology details.
- Richard West provided insight on areas of virtualization research.
- Clay Breshears helped with coverage of Intel thread-building blocks.
- Gerry Howser gave feedback on motivating the study of operating systems and also tried out new material in his class.
- Judi Paige helped with generating figures and presentation of slides.
- Jay Gagne and Audra Rissmeyer prepared new artwork for this edition.
- Owen Galvin provided technical editing for Chapters 11 and 12.
- Mark Wogahn has made sure that the software to produce this book (LaTeX and fonts) works properly.
- Ranjan Kumar Meher rewrote some of the LaTeX software used in the production of this new text.

Previous Editions

- **First three editions**. This book is derived from the previous editions, the first three of which were coauthored by James Peterson.
- **General contributions**. 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, Bart Childs, Ajoy Kumar Datta, Joe Deck, Sudarshan K. Dhall, Thomas Doeppner, Caleb Drake, M. Rasit Eskicioğlu, Hans Flack, Robert Fowler, G. Scott Graham, Richard Guy, Max Hailperin, Rebecca Hartman, Wayne Hathaway, Christopher Haynes, Don Heller, Bruce Hillyer, Mark Holliday, Dean Hougen, Michael Huang, Ahmed Kamel, Morty Kewstel, Richard Kieburtz, Carol Kroll, Morty Kwestel, Thomas LeBlanc, John Leggett, Jerrold Leichter, Ted Leung, Gary Lippman, Carolyn Miller, Michael Molloy, Euripides Montagne, 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, John Sterling, 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.

- **Specific Contributions**
  - Jonathan Katz contributed to Chapter 16. Richard West provided input into Chapter 18. Salahuddin Khan updated Section 16.7 to provide new coverage of Windows 7 security.
  - Parts of Chapter 19 were derived from a paper by Levy and Silberschatz [1990].
  - Cliff Martin helped with updating the UNIX appendix to cover FreeBSD.
  - Some of the exercises and accompanying solutions were supplied by Arvind Krishnamurthy.
  - Andrew DeNicola prepared the student study guide that is available on our website. Some of the slides were prepared by Marilyn Turnamian.
  - Mike Shapiro, Bryan Cantrill, and Jim Mauro answered several Solaris-related questions, and Bryan Cantrill helped with the ZFS coverage. Josh Dees and Rob Reynolds contributed coverage of Microsoft's NET.
  - Owen Galvin helped copy-edit Chapter 18 of this edition.

Book Production

The Executive Editor was Don Fowley. The Senior Production Editor was Ken Santor. The Freelance Developmental Editor was Chris Nelson. The Assistant Developmental Editor was Ryann Dannelly. The cover designer was Tom Nery. The copyeditor was Beverly Peavler. The freelance proofreader was Katrina

Avery. The freelance indexer was WordCo, Inc. The Aptara LaTex team consisted of Neeraj Saxena and Lav Kush.

Personal Notes

Avi would like to acknowledge Valerie for her love, patience, and support during the revision of this book.

Peter would like to thank his wife Carla and his children, Gwen, Owen, and Maddie.

Greg would like to acknowledge the continued support of his family: his wife Pat and sons Thomas and Jay.

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