Preface

Live in fragments no longer. Only connect.
Edward Morgan Forster

Welcome to the world of operating systems. This text is intended primarily for use in the one-semester and two-semester operating systems courses (as defined in the most recent ACM/IEEE curriculum) that universities offer to juniors, seniors and graduate students in computer science. Operating systems designers and systems programmers will find the text useful as a reference as well.

The text features extensive case studies on two of today's most important operating systems—Linux (94 pages) and Microsoft® Windows® XP (106 pages)—that represent two different operating system design paradigms—free, open-source development and licensed, corporate development, respectively. The Linux case study follows the development of the kernel through version 2.6. The Windows XP case study highlights the internals of the current version of the most widely used personal computer operating system. These case studies enable you to compare and contrast the design and implementation philosophies used in real-world operating systems.

Both Linux and Windows XP are massive, complex operating systems containing millions of lines of source code. We survey the major components of each operating system. The case studies present issues in personal computer, workstation, multiprocessor, distributed and embedded environments, including a detailed discussion on why Linux and other UNIX-like operating systems have become prominent in the open-source and open-systems philosophies of major corporations.

This preface introduces our teaching approach in *Operating Systems, 3/e* and the book's key content and design elements. We also discuss the ancillary support available with the text. The section entitled "Tour of the Book" overviews the rich coverage of operating systems this book provides.

Operating Systems, 3/e, was reviewed by a team of distinguished academics and industry professionals; their names and affiliations are listed in the Acknowledgements section.

As you read this book, if you have any questions, please send us an e-mail at **deitel@deitel.com**; we will respond promptly. Please visit our Web site, **www.deitel.com**, regularly and sign up for the *Deitel® Buzz Online* e-mail newsletter at **www.deitel.com/newsletter/subscribe.html**. We use the Web site and the newsletter to keep our readers current on all Deitel publications and services. The site **www.deitel.com/books/os3e** is dedicated to *Operating Systems, 3/e*. At that site, we will post various ancillary materials for students and instructors, including dependency charts, content updates, errata, suggested student research projects (term projects, directed studies and theses), sample syllabi and the bibliography of research materials from the prior edition of the book. Prentice Hall provides a Companion Website for the book at **www.prenhall.com/deitel**.

Book Design

Operating Systems, 3/e features a completely new, two-color design inspired by the work of Leonardo DaVinci. The two-color design is visually appealing and allows us to use color to improve the pedagogy of the book. For example, defining occurrences of key terms appear in **bold and color** so they stand out to you.

Operating Systems, 3/e Features

Operating Systems, 3/e includes extensive new content. We also revised and updated much of the material from the second edition. The focus on current technologies and issues in distributed computing makes this book unique from its competition. Several new sections have been added to address embedded, real-time and distributed systems. Some key features of this edition are:

- Conforms to all core requirements of the ACM/IEEE CC2001 Operating Systems course requirements.
- Discusses all the CC2001 elective operating systems topics except shell scripting.
- Provides an updated hardware introduction that includes leading-edge technologies and their impact on operating systems design.
- Presents process, thread, memory and disk management solutions that reflect the needs of current applications.
- Supplements the extensive coverage of general-purpose systems with concepts relevant to real-time, embedded and superscalar architectures.
- Highlights key evaluation techniques that enable effective comparative analyses of operating system components.
- Includes a richer treatment of networking concepts.
- Enhances the security treatment to include current trends in authentication mechanisms, security protocols, anti-virus research, access control methods and wireless security.

- Enhances the distributed computing coverage to acknowledge the tremendous influence of the Internet and the World Wide Web on computing and operating systems.

- Provides details of the ubiquitous Intel® architecture.

- Provides many diagrams, tables, working code examples, pseudocode examples and algorithms.

- Includes a new chapter on threads.

- Pseudocode is presented in Java-like syntax to capitalize on C/C++/Java literacy—virtually all computer-science students know one or more of these languages.

- Provides multithreading treatments in both pseudocode and Java that demonstrate issues in concurrent programming—enabling instructors to cover the material the way they prefer. The Java treatment is new to this edition and is optional. Visit `java.sun.com/j2se/downloads.html` to obtain the latest version of Java. The download page contains a link to the installation instructions.

- Enhances the multiprocessor management treatment.

- Provides new sections on thread scheduling and real-time scheduling.

- Includes a RAID discussion.

- Provides a case study on UNIX processes.

- Includes up-to-the-minute sections on memory-management and disk-scheduling strategies.

- Covers the important topic of I/O systems in many chapters—most notably Chapters 2, 12, 13 and the case study chapters (20 and 21).

- Provides 730 self-review questions and answers (approximately two per section) for immediate feedback.

- Includes extensive research with citations listed in the Works Cited section at the end of each chapter.

Teaching Approach

This book is divided into eight parts, each containing a set of related chapters. The parts are:

1. Introduction to Hardware, Software and Operating Systems

2. Processes and Threads

3. Real and Virtual Memory

4. Secondary Storage, Files and Databases

5. Performance, Processors and Multiprocessor Management

6. Networking and Distributed Computing

7. Security
8. Case Studies

The book's pedagogic features are described below.

Quotations

Each chapter begins with quotations—some are humorous, some are philosophical and some offer interesting insights. Many readers have told us that they enjoy relating the quotations to the chapter material. You might appreciate some of the quotations more *after* reading the chapters.

Objectives

Next, objectives tell you what to expect and give you an opportunity, after reading the chapter, to determine whether you have met these objectives.

Outline

The chapter outline helps you approach the material in a top-down fashion, so you can anticipate the concepts that will be presented in each chapter and set a comfortable learning pace.

Sections and Self-Review Exercises

Each chapter contains small sections that address important operating systems concepts. Most sections end with two self-review exercises and answers. These exercises enable you to test your knowledge, get immediate feedback and gauge your understanding of the material. These exercises also help prepare you for the end-of-chapter exercises and for quizzes and exams. Some of the self-review exercises cannot be answered only from the material presented in their corresponding sections; these are additional teaching and learning opportunities.

Key Terms

The defining occurrence of each term appears in **bold and color**. In addition, all chapters include a Key Terms section containing the terms defined in the chapter and their definitions (1800+ key terms in the text). A cumulative alphabetized glossary appears at the end of the book. The Key Terms sections and the Glossary are wonderful pedagogic devices for review and reference.

Figures

The text contains over 300 charts, diagrams, examples and illustrations that support the concepts presented in the text.

Web Resources

Each chapter contains Web resources that direct you to sites where you can locate valuable additional research materials.

Summary

Detailed end-of-chapter summary sections help you review the key concepts presented in each chapter.

Exercises, Suggested Projects and Suggested Simulations

Each chapter includes many exercises that vary in difficulty from review of basic operating systems principles to complex reasoning and research projects (900+ exercises in the text). Many OS instructors like to assign term projects, so we have included Suggested Projects and Suggested Simulations sections at the end of each chapter's exercises.

Recommended Readings

Each chapter lists recommended books and articles, and provides a brief review of the most important sources for the chapter content so you can do additional research on your own.

Works Cited

This book required an extraordinary research effort. The entire book is thoroughly cited (2300+ citations). Each citation appears as a superscript in the text and corresponds to an entry in the Works Cited section at the end of the chapter. Many of these citations are to Web sites. Prior editions of this text used only standard book and literature citations; those books and papers were often difficult for readers to locate for further research. Now you can access these citations directly via the Web. Also, it is easy to use search engines to locate additional articles on subjects of interest. Many research journals are accessible online—some are free and others are available through personal or organizational memberships in professional societies. The Web is truly a research bonanza that leverages your learning experience.

Index

We provide a two-level index to help you locate any term or concept quickly. For example, each operating system mentioned in the text appears in the index both alphabetically and indented under the term "Operating Systems."

Feature Boxes

The second edition of *Operating Systems* included a full chapter on analytic modeling with queuing theory and Markov processes. This edition omits that material, recognizing that operating systems is for the most part not a mathematical field. Rather it has a basis in what we call "systems thinking"—operating systems is largely a field of empirical results. To illuminate these issues, we have included in this edition four types of feature boxes, presenting material to challenge, entertain and enrich you.

Biographical Notes

Eighteen Biographical Notes provide interesting details about people who have made significant contributions to the field of operating systems—*Edsger Dijkstra*, *Linus Torvalds, David Cutler, Ken Thompson, Dennis Ritchie, Doug Engelbart, Tim Berners-Lee, Richard Stallman, Gordon Moore, Fernando J. Corbató, Leslie Lamport, Per Brinch Hansen, Peter Denning, Seymour Cray, Bill Gates, Ronald Rivest, Adi Shamir* and *Leonard Adleman*. Please do not attach any particular significance to our choices; these are only a few among thousands of important contributors. Please let us know of other people you feel deserve mention.

Mini Case Studies

In addition to the detailed case studies on the Linux and Windows XP operating systems, 14 Mini Case Studies focus on other important operating systems of historical research or commercial interest. These Mini Case Studies include *Mach, CTSS and Multics, UNIX Systems, Real-Time Operating Systems, Atlas, IBM Mainframe Operating Systems, Early History of the VM Operating System, MS-DOS, Supercomputers, Symbian OS, OpenBSD, Macintosh, User-Mode Linux (UML)* and *OS/2*.

Anecdotes

One of the authors, HMD, accumulated many anecdotes over decades of teaching operating systems in academia and industry. We have included 16 of these anecdotes. Some are humorous; others are thought provoking—raising deep philosophical issues. Each is a (hopefully) pleasant diversion from the book's technical discussions and concludes with a *Lesson to operating systems designers*.

Operating Systems Thinking

Academics enjoy the luxury of being able to study what is interesting about operating systems, especially clever algorithms, data structures and occasionally, areas that lend themselves nicely to mathematical analysis. Industry professionals must build real systems that work and meet the demanding cost, performance and reliability requirements of customers. Both kinds of thinking are rich with interesting issues. There are considerable overlaps as well as significant differences in what academics and industry professionals think about. This book aims to present a balanced treatment of both the academic and industry sides of operating systems theory and practice.

What is "operating systems thinking?" Forty-three Operating Systems Thinking features explore that question. Indeed, some aspects of operating systems lend themselves to sophisticated mathematical analysis. But the author's (HMD's) extended experience in the computer industry—42 years, including working (as a very junior person) on major operating systems research and development efforts at IBM and MIT, writing two earlier editions of this book and teaching operating systems in academia and industry dozens of times—has shown that these systems are far too complex for a significant mathematical treatment at the undergraduate or early graduate level. Even at the advanced graduate level, except for narrow areas of interest, operating systems defy mathematical analysis.

If there is not a mathematical basis for evaluating aspects of operating systems, how can we think about them productively? The answer is what we call "systems thinking," and the text certainly covers this. However, the Operating Systems Thinking features are our effort to capture key concepts that are prevalent in operating systems design and implementation.

The Operating Systems Thinking features are: *Innovation*; *Relative Value of Human and Computer Resources*; *Performance*; *Keep It Simple (KIS)*; *Architecture*; *Caching*; *Legacy Hardware and Software*; *Principle of Least Privilege*; *Protection*; *Heuristics*; *Customers Ultimately Want Applications*; *Data Structures in Operating Systems*; *Asynchronism vs. Synchronism*; *Concurrency*; *Parallelism*; *Standards Conformance*; *Scalability*; *Information Hiding*; *Waiting, Deadlock and Indefinite Postponement*; *Overhead*; *Predictability*; *Fairness*; *Intensity of Resource Management vs. Relative Resource Value*; *There Are No Upper Limits to Processing Power, Memory, Storage and Bandwidth*; *Change Is the Rule Rather Than the Exception*; *Spatial Resources and Fragmentation*; *Virtualization*; *Empirical Results and Locality-Based Heuristics*; *Lazy Allocation*; *Computer Theory in Operating Systems*; *Space–Time Trade-offs*; *Saturation and Bottlenecks*; *Compression and Decompression*; *Redundancy*; *Fault Tolerance*; *Mission-Critical Systems*; *Encryption and Decryption*; *Security*; *Backup and Recovery*; *Murphy's Law and Robust Systems*; *Graceful Degradation*; *Data Replication and Coherency* and *Ethical Systems Design*

Case Studies

Chapters 20 and 21 cover in depth the Linux and Windows XP operating systems, respectively. These thorough case studies were carefully reviewed by key Linux and Windows XP operating systems developers. The outlines for each of these case studies mimic the text's table of contents. The case studies truly reinforce the text's key concepts—the text presents the principles; the case studies show how these principles are applied in building today's two most widely used operating systems. The Linux case study follows the development of the latest kernel release (v. 2.6) and contains 262 citations. The Windows XP case study reflects the latest Windows operating system features and contains 485 citations.

Tour of the Book

This section provides an overview of the eight parts and 21 chapters of *Operating Systems, 3/e.*

Part 1—Introduction to Hardware, Software and Operating Systems— includes two chapters that introduce the notion of operating systems, present a history of operating systems and lay the groundwork of hardware and software concepts the reader will use throughout the book.

Chapter 1—Introduction to Operating Systems— defines the term "operating system" and explains the need for such systems. The chapter provides a historical perspective on operating systems, tracing their development decade by decade

through the second half of the 20th century. The batch-processing systems of the 1950s are considered. We see the 1960s trend towards parallelism with the advent of mutiprogramming—both batch mutiprogramnming and interactive timesharing systems. We follow the development of key operating systems including CTSS, Multics, CP/CMS and Unix. The chapter considers the kinds of thinking that operating systems designers did in an era when computing resources were far more expensive than people resources (today the reverse is true). We follow the 1970s evolution of computer networking, the Internet and the TCP/IP protocol suite, and see the beginnings of the personal computing revolution. Personal computing matures in the 1980s with the release of the IBM personal computer and the Apple Macintosh, the latter popularizing the graphical user interface (GUI). We see the beginnings of distributed computing and the development of the client/server model. In the 1990s, Internet usage literally explodes with the availability of the World Wide Web. Microsoft becomes the world's dominant software maker and releases its Windows NT operating system (the ancestor of today's Windows XP operating system—the focus of Chapter 21). Object technology becomes the dominant development paradigm with languages like C++ and Java becoming popular. The rapid rise of the open-source-software movement leads to the phenomenal success of the Linux operating system—the focus of Chapter 20. We discuss how operating systems provide a platform for applications development. Embedded systems are considered with an emphasis on how crucial it is that mission-critical and business-critical systems be extraordinarily reliable. We consider the core components of operating systems and key operating systems goals. (The book maintains a focus on performance issues in virtually every aspect of operating systems.) Operating system architectures are introduced, including monolithic architecture, layered architecture, microkernel architecture and networked and distributed operating systems.

Chapter 2—Hardware and Software Concepts—summarizes the hardware and software resources operating systems manage. The chapter covers how trends in hardware design—most notably phenomenal increases in processing power, memory capacity and communications bandwidth—have affected operating system design and vice versa. Hardware components including mainboards, processors, clocks, main memory, secondary storage devices, buses, direct memory access (DMA), peripheral devices and more are discussed. We present recent and emerging hardware technologies, and discuss hardware support for operating systems, including processor execution modes, privileged instructions, timers, clocks, bootstrapping and plug-and-play. Performance enhancement techniques like caching and buffering are considered. Software concepts are examined including compiling, linking, loading, machine languages, assembly languages, interpreters, compilers, high-level languages, structured programming, object-oriented programming and application programming interfaces (APIs). The chapter also explains firmware and middleware.

Part 2—Processes and Threads—includes six chapters that present the notions of processes, threads, process- and thread-state transitions, interrupts, con-

text switching, asynchronism, mutual exclusion, monitors, deadlock and indefinite postponement, and processor scheduling of processes and threads.

Chapter 3—Process Concepts—begins our discussion of operating system primitives by defining the fundamental notion of process. We consider the life cycle of a process as it transitions between process states. The representation of a process as a process-control block or process descriptor is discussed with an emphasis on the importance of data structures in operating systems. The chapter motivates the need for process structures in an operating system and describes operations that can be performed on them, such as suspend and resume. Multiprogramming considerations, including suspending process execution and context switching, are introduced. The chapter discusses interrupts—a key to the successful implementation of any multiprogrammed environment. We consider interrupt processing and interrupt classes, interprocess communication with signals and message passing. We conclude with a case study of UNIX processes.

Chapter 4—Thread Concepts—extends our discussion of process concepts to a smaller unit of concurrent program execution: the thread. The chapter defines what threads are and explains their relationship to processes. The life cycle of a thread and how threads transition among various thread states is discussed. We consider various threading architectural models, including user-level threads, kernel-level threads and combining user- and kernel-level threads. The chapter presents thread implementation considerations, including thread signal delivery and thread termination. We discuss the POSIX standard and its threading specification, Pthreads. The chapter concludes with a presentation of the Linux, Windows XP and Java threading implementations. The Java presentation is accompanied by a complete, working Java program with sample outputs. [*Note:* The Java source code for all of the Java programs in the book is available for download at `www.deitel.com/books/os3e`.]

Chapter 5—Asynchronous Concurrent Execution—discusses the issues of concurrency encountered in multiprogrammed systems. The chapter introduces the problem of mutual exclusion and how threads must manage access to shared resources. A feature of the chapter is the Java Multithreading Case Study: A Producer/Consumer Relationship in Java—which uses a complete, working Java program and several sample outputs to clearly illustrate what happens when concurrent threads access shared data without synchronization. This example clearly shows that sometimes such a concurrent program will work fine and sometimes it will produce erroneous results. We show how to solve this problem with a multithreaded Java program in Chapter 6. The concept of the critical section in program code is introduced. Several software mechanisms that protect access to critical sections are presented as solutions to the mutual exclusion problem; these include Dekker's Algorithm, Peterson's Algorithm and *N*-thread mutual exclusion with Lamport's Bakery Algorithm. The chapter also discusses hardware mechanisms that facilitate the implementation of mutual exclusion algorithms; these include disabling interrupts, the test-and-set instruction and the swap instruction. Finally, semaphores are presented as a high-level mechanism for

implementing mutual exclusion and thread synchronization; both binary semaphores and counting semaphores are considered.

Chapter 6—Concurrent Programming—explains the notion of monitors (a high-level mutual exclusion construct) then proceeds to solve several classic problems in concurrent programming, using first pseudocode monitors then complete working Java programs with sample outputs. The monitor solutions are provided in a C/C++/Java-like pseudocode syntax. We explain how monitors enforce information hiding and discuss how monitor condition variables differ from "conventional" variables. The chapter illustrates how a simple monitor may be used to control access to a resource that requires exclusive use. We then discuss two classic problems in concurrent programming—the circular buffer and the readers-and-writers problem; we implement solutions to each of these with pseudocode monitors. Java monitors are explained and the differences between Java monitors and the ones described by the classic literature are discussed. The chapter continues our optional Java Multithreading Case Study by implementing a producer/consumer relationship and a circular buffer in Java. The student can use this latter program as a basis for implementing a solution to the readers-and-writers problem in Java.

Chapter 7—Deadlock and Indefinite Postponement—introduces two potentially disastrous consequences of waiting: deadlock and indefinite postponement. The key concern is that systems that manage waiting entities must be carefully designed to avoid these problems. Several examples of deadlock are presented, including a traffic deadlock, the classic one-lane bridge, a simple resource deadlock, deadlock in spooling systems and deadlock in Dijkstra's charming Dining Philosophers problem. Key resource concepts are considered, including preemptibility, sharing, reentrancy and serial reusability. The chapter formally defines deadlock and discusses deadlock prevention, avoidance, detection and recovery (almost invariably painful). The four necessary conditions for deadlock are explained, namely the "mutual-exclusion," "wait-for," "no-preemption" and "circular-wait" conditions. We examine Havender's methods for preventing deadlock by denying, individually, any of the last three conditions. Deadlock avoidance enables more flexible resource allocation than deadlock prevention. The chapter explains deadlock avoidance with Dijkstra's Banker's Algorithm, showing examples of a safe state, an unsafe state and a safe-state-to-unsafe-state transition. Weaknesses in the Banker's Algorithm are discussed. We explain deadlock detection with the resource-allocation graph reduction technique. The chapter concludes with a discussion of deadlock strategies in current and future systems.

Chapter 8—Processor Scheduling—discusses concepts and algorithms related to allocating processor time to processes and threads. Scheduling levels, objectives and criteria are considered. Preemptive and nonpreemptive scheduling approaches are compared. We explain how to carefully set priorities and quanta (finite-sized allocations of processor time) in scheduling algorithms. Several classic and current scheduling algorithms are considered, including first-in-first-out (FIFO), round-robin (RR), shortest-process-first (SPF), highest-response-ratio-next (HRRN),

shortest-remaining-time (SRT), multilevel feedback queues and fair-share scheduling. Each algorithm is evaluated using metrics such as throughput, average response time and the variance of response times. We discuss Java thread scheduling, soft real-time scheduling, hard real-time scheduling and deadline scheduling.

Part 3—Real and Virtual Memory—includes three chapters that discuss memory organization and memory management in real and virtual memory systems.

Chapter 9—Real Memory Management and Organization—presents a historical discussion of how real memory operating systems have organized and managed physical memory resources. The schemes have gone from the simple to the complex, ultimately seeking optimal usage of the relatively precious main memory resource. We review the memory hierarchy consisting of cache(s), primary memory and secondary storage. Then, three types of memory management strategies are discussed, namely fetch, placement and replacement. We present contiguous and noncontiguous memory allocation schemes. Single-user contiguous memory allocation is considered with a discussion of overlays, protection and single-stream batch processing. We trace the evolution of multiprogramming memory organizations from fixed-partition multiprogramming to variable-partition multiprogramming, considering issues including internal and external memory fragmentation, and presenting memory compaction and coalescing as means of reducing fragmentation. The chapter discusses the first-fit, best-fit and worst-fit memory placement strategies, and concludes with a discussion of multiprogramming with memory swapping.

Chapter 10—Virtual Memory Organization—describes fundamental virtual memory concepts and the hardware capabilities that support virtual memory. The chapter motivates the need for virtual memory and describes typical implementations. The key approaches to virtual memory organization—paging and segmentation—are explained and their relative merits are analyzed. We discuss paging systems, focusing on paging address translation by direct mapping, paging address translation by associative mapping, paging address translation with direct/associative mapping, multilevel page tables, inverted page tables and sharing in paging systems. The chapter investigates segmentation systems, focusing on segmentation address translation by direct mapping, sharing in a segmentation system, protection and access control in segmentation systems. We also examine hybridized segmentation/paging systems, considering dynamic address translation, sharing and protection in these systems. The chapter concludes by examining the popular IA-32 Intel architecture virtual-memory implementation.

Chapter 11—Virtual Memory Management—continues the discussion of virtual memory by analyzing how operating systems attempt to optimize virtual memory performance. Because paging systems have become dominant, we focus on page management in detail, most notably on page-replacement strategies. The chapter considers one of the most important empirical results in the field of operating systems, namely the locality phenomenon, and we consider the results from both the temporal and spatial perspectives. We discuss when pages should be brought into memory, examining both demand paging and anticipatory paging.

When available memory becomes scarce, incoming pages must replace pages already in memory—an operating system's page-replacement strategy can have an enormous impact on performance. Many page-replacement strategies are examined, including random, first-in-first-out (FIFO and Belady's Anomaly), least-recently-used (LRU), least-frequently-used (LFU), not-used-recently (NUR), second-chance, clock and segmented queue (SEGQ), far page and page-fault-frequency (PFF). Denning's classic working set model of program behavior is considered as well as how various page-replacement strategies attempt to achieve its goals. The chapter discusses the possibility of voluntary page release. We carefully examine the arguments for small pages and large pages, and how programs behave under paging. The chapter discusses Linux page replacement and concludes with a discussion of global vs. local page-replacement strategies.

Part 4—Secondary Storage, Files and Databases—includes two chapters that present techniques which operating systems employ to manage data on secondary storage. Hard-disk performance optimization, and file and database systems are discussed. Our treatment of I/O systems is distributed throughout the book, most notably in Chapters 2, 12 and 13 and in the case studies on Linux and Windows XP (Chapters 20 and 21, respectively).

Chapter 12—Disk Performance Optimization—focuses on the characteristics of moving-head disk storage and how the operating system can optimize its performance. The chapter discusses the evolution of secondary-storage devices and examines the characteristics of moving-head disk storage. During a half century of explosive technological development, moving-head disk storage devices continue to endure. We define why disk scheduling is necessary and demonstrate why it is needed to achieve high-performance from moving-head disk devices. The evolution of disk-scheduling strategies is presented, including seek-optimization strategies, including first-come-first-served (FCFS), shortest-seek-time-first (SSTF), SCAN, C-SCAN, FSCAN, N-Step SCAN, LOOK, C-LOOK and VSCAN (introduced in the exercises), and rotational-optimization strategies, including SLTF, SPTF and SATF. Other popular disk-system performance techniques are discussed, including caching, buffering, defragmentation, data compression and blocking. One of the most important additions to this third edition is the extensive treatment of RAID systems in this chapter. RAID (Redundant Arrays of Independent Disks) is a set of technologies that enable disk systems to achieve higher performance and fault tolerance. The chapter presents various key RAID "levels," including level 0 (striping), level 1 (mirroring), level 2 (bit-level Hamming ECC parity), level 3 (bit-level XOR ECC parity), level 4 (block-level XOR ECC parity) and level 5 (block-level distributed XOR ECC parity).

Chapter 13—File and Database Systems—discusses how operating systems organize and manage collections of data called files and databases. We explain key concepts, including the data hierarchy, files, file systems, directories, links, metadata and mounting. The chapter presents various file organizations, including sequential, direct, index sequential and partitioned. File allocation techniques are examined, including contiguous file allocation, linked-list noncontiguous file allocation, tabu-

lar noncontiguous file allocation and indexed noncontiguous file allocation. We explain file access control by user classes and access control matrices. Data access techniques are discussed in the context of basic access methods, queued access methods, anticipatory buffering and memory-mapped files. We explain techniques for ensuring data integrity, including protection, backup, recovery, logging, atomic transactions, rollback, commitment, checkpointing, shadow paging and using log-structured file systems. The chapter considers file servers and distributed systems and points the reader to the extensive treatment of these topics and others in Chapters 16–18. The chapter also introduces database systems, and analyzes their advantages, data access, the relational database model and operating system services that support database systems.

Part 5—Performance, Processors and Multiprocessor Management—includes two chapters that discuss performance monitoring, measurement and evaluation techniques, and focus on the extraordinary performance that can be achieved with systems that employ multiple processors.

Chapter 14—Performance and Processor Design—focuses on one of operating systems designers' most preeminent goals—system performance—and discusses the important role of specific processor types in achieving that goal. The chapter surveys performance measures, considering issues including absolute performance measures, ease of use, turnaround time, response time, system reaction time, variance in response times, throughput, workload, capacity and utilization. We discuss performance evaluation techniques, including tracing and profiling, timings, microbenchmarks, application-specific evaluation, analytic models, benchmarks, synthetic programs, simulation and performance monitoring. Bottlenecks and saturation are considered. We demonstrate how systems dynamically adjust to positive and negative feedback. A system's performance depends heavily on the performance of its processor(s), which is, in turn, heavily influenced by its instruction set architecture. The chapter examines key architectures, including complex instruction set computing (CISC), reduced instruction set computing (RISC) and various kinds of post-RISC processors. The chapter concludes with a discussion of explicitly parallel instruction computing (EPIC).

Chapter 15—Multiprocessor Management—provides an in-depth introduction to the hardware and software aspects of multiprocessing. One way to build ever more powerful computing systems is to employ multiple processors and possibly massive numbers of them. The discussion begins with a Mini Case Study on supercomputers followed by a Biographical Note on Seymour Cray, the father of supercomputing. We investigate multiprocessor architecture, considering issues including the classification of sequential and parallel architectures, processor interconnection schemes and loosely coupled vs. tightly coupled systems. Multiprocessor operating system organizations are examined, including master/slave, separate kernels and symmetrical organization. We explain memory access architectures, including uniform memory access (UMA), nonuniform memory access (NUMA), cache-only memory architecture (COMA) and no remote memory access (NORMA).

The chapter discusses multiprocessor memory sharing, considering issues of cache coherence, page replication and migration and shared virtual memory. We continue the discussion on processor scheduling which we began in Chapter 8, presenting job-blind multiprocessor scheduling and job-aware multiprocessor scheduling. Process migration is considered and issues of flow of process migration and process migration strategies are examined. We discuss multiprocessor load balancing, examining both static and dynamic load balancing strategies. The chapter explains how to enforce multiprocessor mutual exclusion with spin locks, sleep/wakeup locks and read/write locks.

Part 6—Networking and Distributed Computing—includes three chapters that present networks, networked systems and distributed systems.

Chapter 16—Introduction to Networking—introduces computer networking to lay the foundation for the following two chapters on distributed computing. The chapter discusses network topologies, including buses, rings, meshes, fully connected meshes, stars and trees and we explain the unique challenges posed by wireless networks. We consider local-area networks (LANs) and wide-area networks (WANs), and present a major treatment of the TCP/IP protocol stack. Application layer protocols including the Hypertext Transfer protocol (HTTP) and the File Transfer Protocol (FTP) are examined. We explain transport layer protocols including the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). At the network layer the chapter explores the Internet Protocol (IP) and its latest version—Internet Protocol version 6 (IPv6). The chapter discusses link layer protocols, including Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI) and IEEE 802.11 (wireless). The chapter concludes with a discussion of the client/server model and *n*-tier systems.

Chapter 17—Introduction to Distributed Systems—introduces distributed operating systems and discusses the attributes of distributed systems, including performance, scalability, connectivity, security, reliability, fault tolerance and transparency. We compare and contrast network operating systems and distributed operating systems. Communication in distributed systems and the crucial role played by "middleware" technologies are discussed, including remote procedure call (RPC), Remote Method Invocation (RMI), CORBA (Common Object Request Broker Architecture), and Microsoft's DCOM (Distributed Component Object Model). The chapter considers process migration, synchronization and mutual exclusion in distributed systems, examining mutual exclusion without shared memory, and Agrawala and Ricart's distributed mutual exclusion algorithm. We discuss deadlock in distributed systems, focusing on deadlock prevention and deadlock detection. The chapter concludes with case studies on the Sprite and Amoeba distributed operating systems.

Chapter 18—Distributed Systems and Web Services—continues the study of distributed systems, focusing on distributed file systems, clustering, peer-to-peer computing, grid computing, Java distributed computing and Web services. The chapter begins by considering some characteristics and concerns of distributed systems,

including transparency, scalability, security, fault tolerance and consistency and we present case studies on key distributed file systems, including the Network File System (NFS), Andrew File System (AFS), Coda File System and Sprite File System. We discuss clustering, considering high-performance clusters, high-availability clusters and load-balancing clusters; we investigate various examples of clusters including Linux-based Beowulf clusters Windows-based clusters. Peer-to-peer (P2P) distributed computing is explored considering the relationship between P2P and client/server applications, centralized vs. decentralized P2P applications, peer discovery and searching. We examine Sun Microsystems' Project JXTA and we consider how JXTA creates a framework for building P2P applications. The chapter considers grid computing and how it makes possible solutions to problems that require a truly massive amount of computation by using available unused processing power on personal and business computers worldwide. We discuss key Java distributed computing technologies, including Java servlets and JavaServer Pages (JSP), Jini, JavaSpaces and Java Management Extensions (JMX). The chapter concludes by surveying the exciting new technology of Web services and exploring two key Web services platforms—Microsoft's .NET Platform and Sun Microsystems' Sun ONE platform.

Part 7—Security—includes one chapter.

Chapter 19—Security—presents a general introduction to computer and network security techniques that operating systems can use to provide secure computing environments. The chapter discusses secret-key and public-key cryptography and the popular RSA and PGP (Pretty Good Privacy) public key schemes. Authentication is considered: password protection, password salting, biometrics, smart cards. Kerberos and single sign-on. We discuss access control, exploring issues of access rights, protection domains, access control models, access control policies and access control mechanisms, including access control matrices, access control lists and capability lists. The chapter presents the great variety of security attacks that have been attempted, including cryptanalysis, viruses, worms, denial-of-service (DoS) attacks, software exploitation and system penetration. We survey attack prevention and security solutions, including firewalls and intrusion detection systems (IDSs), antivirus software, security patches and secure file systems. We discuss the federal government's Orange Book Security classification system and present a Mini Case Study on OpenBSD, arguably the most secure operating system available. The chapter explores secure communication and consider the requirements for a successful, secure transaction—privacy, integrity, authentication, authorization and nonrepudiation. Key agreement protocols—the process by which two parties exchange secret keys over an unsecure medium—are discussed. We explain digital signatures—a technology that is enormously crucial to the future of electronic commerce—and their implementation. The chapter presents an in-depth treatment of public-key infrastructure, including digital certificates and certificate authorities. Various secure communication protocols are considered including Secure Sockets Layer (SSL), Virtual Private Networks (VPNs), IP Security (IPSec) and wireless security. We discuss the intriguing technology of steganography—the practice of

hiding information within other information. This can hide secret messages in publicly transmitted messages; it can also protect intellectual property rights, with digital watermarks for example. The chapter compares and contrasts proprietary and open-source security solutions and concludes with a case study on UNIX Security.

Part 8—Operating Systems Case Studies—includes two chapters that provide in-depth case studies on the Linux 2.6 kernel and the Microsoft Windows XP operating system,. The case studies follow the book's outline to make it convenient for the student to comprehend topics discussed earlier in the book.

Chapter 20—Case Study: Linux—provides an in-depth study of Linux 2.6. This extensive (94 pages) case study was written and updated throughout the development of the release (by following the version 2.5 development kernel and having the material carefully reviewed by key Linux kernel developers). The chapter discusses the history, community and software distributions that have created the most popular open-source operating system in the world. The chapter examines the core components of the Linux operating system, with particular attention to their implementation in the context of concepts studied in previous chapters. Kernel architecture, process management (processes, threads, scheduling), memory organization and management, file systems (virtual file system, virtual file system caches, ext2fs and proc fs), I/O management (device drivers, character device I/O, block device I/O, network device I/O, unified device model, interrupts), synchronization (spin locks, reader/writer locks, seqlocks and kernel semaphores), IPC (signals, pipes, sockets, message queues, shared memory and System V semaphores), networking (packet processing, netfilter framework and hooks), scalability (symmetric multiprocessing, nonuniform memory access, other scalability features, embedded Linux) and security are all analyzed and explained.

Chapter 21—Case Study: Windows XP—complements the Linux case study by examining the internals of the most popular commercial operating system—Windows XP. This case study (106 pages) examines the core components of the Windows XP operating system and how they interact to provide services to users. We discuss the history of Windows operating systems, including Biographical notes on Bill Gates and David Cutler. The chapter presents the design goals of Windows XP and overviews its system architecture, considering topics including the Hardware Abstraction Layer (HAL), the microkernel, the executive, the environment subsystems, dynamic link libraries (DLLs) and system services. We discuss system management mechanisms, including the registry, Object Manager, interrupt request levels (IRQLs), asynchronous procedure calls (APCs), deferred procedure calls (DPCs) and system threads. We examine process and thread organization, considering control blocks, thread local storage (TLS), creating and terminating processes, jobs, fibers and thread pools. Thread scheduling is explained, and thread states, the thread scheduling algorithm, determining thread priorities and multiprocessor scheduling are considered. We investigate thread synchronization, examining dispatcher objects, event objects, mutex objects, semaphore objects, waitable timer objects, kernel mode locks and other synchronization tools. Memory management,

and the concepts of memory organization, memory allocation and page replacement are explained. We explore file systems management, discussing file system drivers and NTFS topics including the Master File Table (MFT), data streams, file compression, file encryption, sparse files, reparse points and mounted volumes. We study input/output management, explaining device drivers, Plug-and-Play, power management, the Windows Driver Model (WDM), input/output processing, I/O request packets (IORPs), synchronous I/O, asynchronous I/O, data transfer techniques, interrupt handling and file cache management. We consider interprocess communication mechanisms including pipes (anonymous and named), mailslots, shared memory, and local and remote procedure calls. Microsoft's Component Object Model (COM) is overviewed. We explain drag-and-drop and compound documents. We discuss networking capabilities, including network input/output, network driver architecture, network protocols (IPX, SPX, NetBEUI, NetBIOS over TCP/IP, WinHTTP, WinINet and Winsock 2. Network services topics including Active Directory, Lightweight Directory Access Protocol (LDAP) and Remote Access Service (RAS) are discussed. We overview Microsoft's new .NET technology which is replacing DCOM. We examine scalability, considering both symmetric multiprocessing (SMP) and Windows XP Embedded. The chapter concludes with a discussion of security topics, including authentication, authorization, Internet Connection Firewall (ICF) and other security features.

Whew! Well, that completes the Tour of the Book. There is much here for operating systems students and professionals.

Ancillary Package for Operating Systems, 3/e

Operating Systems, 3/e has extensive ancillary materials for instructors. The Instructor's Resource CD (IRCD) contains solutions to the vast majority of the end-of-chapter exercises. This CD is available only to instructors through their Prentice Hall representatives. Visit **vig.prenhall.com/replocator** to locate your Prentice Hall representative. [*NOTE:* **Please do not write to us requesting the instructor's CD. Distribution of this CD is limited strictly to college professors teaching from the book. Instructors may obtain the solutions manual only from their Prentice Hall representatives.]** The ancillaries for the book also include a Test Item File of multiple-choice questions. In addition, we provide PowerPoint® lecture notes containing all figures, illustrations, diagrams and code in the text and bullet points that summarize key portions of the text. Instructors can customize the slides. The PowerPoint® slides are downloadable from **www.deitel.com** and as part of Prentice Hall's companion Web site (**www.prenhall.com/deitel**) for *Operating Systems, 3/e*, which offers resources for both instructors and students. For instructors, the Companion Web site includes a Syllabus Manager, which helps instructors plan courses interactively and create online syllabi.

Students also benefit from the functionality of the Companion Web site. Book-specific resources for students include:

- Customizable PowerPoint® lecture notes
- Source code for the Java programs

Chapter-specific resources available for students include:

- Chapter objectives
- Outline
- Highlights (e.g., chapter summary)
- Web links

Acknowledgments

We would like to thank the design team that created a completely new look and feel for *Operating Systems, 3/e*—Carole Anson, Paul Belfanti, Geoffrey Cassar and John Root. We would also like to thank the talented creative team at Artworks, including Kathryn Anderson, Jay McElroy, Ronda Whitson, Patricia Burns, Matt Haas, Xiaohong Zhu, Dan Missildine, Chad Baker, Sean Hogan, Audrey Simonetti, Mark Landis, Royce Copenheaver, Stacy Smith, Scott Wieber, Pam Taylor, Anna Whalen, Cathy, Shelly, Ken Mooney, Tim Nguyen, Carl Smith, Jo Thompson, Helenita Zeigler and Russ Crenshaw.

The most important acknowledgment is to the thousands of authors represented in the Recommended Readings and Works Cited sections in the chapters; their research papers, articles and books have provided the diversity of interesting material that makes operating systems such a fascinating area.

We wish to acknowledge the efforts of our *Third Edition* reviewers and to give a special note of thanks to Carole Snyder and Jennifer Cappello of Prentice Hall who managed this extraordinary review effort. Adhering to a tight time schedule, these reviewers scrutinized the text, providing countless suggestions for improving the accuracy and completeness of the presentation. It is a privilege to have the guidance of such talented and busy professionals.

Euripides Montagne, University of Central Florida
Andrew Morton, Diego, Inc.
Gary Newell, Northern Kentucky University
Bill O'Farrell, IBM
Mike Panetta, Robotic Sciences, Inc./Netplanner Systems, Inc.
Rohan Phillips, Microsoft
Atul Prakash, University of Michigan
Bina Ramamurthy, SUNY Buffalo
Eric Raymond, Thyrsus Enterprises
Jeffrey Richter, Wintellect
Sven Schreiber, Platon Data Technology
Wennie Wei Shu, University of New Mexico
David Solomon, David Solomon Expert Seminars
Brandon Taylor, Sun Microsystems
Bob Toxen, Fly-By-Day Consulting, Inc.
Joshua Uziel, Sun Microsystems
Carlos Valcarcel, EinTech, Inc.
Melinda Ward, Hewlett-Packard
Gerard Weatherby, Charles Consulting, LLC. /Rensselaer at Hartford
Xiang Xu, RSA Security, Inc.

First and Second Edition reviewers and contributors
(and their affiliations at that time):
Howard Berkowitz, Corporation for Open Systems
Dale A. Brown, The College of Wooster
Steven J. Buroff, AT&T Bell Laboratories
David D. Busch
John H. Carson, George Washington University
Ronald Curtis, Canisius College
Larry K. Flanigan, University of Michigan
Jon Forrest, Sybase
Jay Gadre, Corporation for Open Systems
Carlos M. Gonzalez, George Mason University
Mike Halvorson, Microsoft Press
Wayne Hathaway, Ultra Network Technologies
Christopher T. Haynes, Indiana University
Lee Hollaar, University of Utah
William Horst, Corporation for Open Systems
James Johannes, University of Alabama
Ralph Johnson, University of Illinois
Dennis Kafura, Virginia Polytechnical Institute
Herb Klein, Corporation for Open Systems
Michael S. Kogan, IBM
Thomas LeBlanc, University of Rochester

T. F. Leibfried, University of Houston, Clear Lake
David Litwack, Corporation for Open Systems
Mark Measures, Baylor University
Charles Oualline, Jr., East Texas State University
Ravi Sandhu, Ohio State University
Richard Schlichting, University of Arizona
Alan Southerton
John J. Zenor, California State University
James Peterson, University of Texas at Austin
Richard Wexelblatt, Sperry Univac
Paul Ross, Millersville State University
Anthony Lucido, Intercomp
Steve Paris, Prime Computer
Bart Guerreri, DSD Laboratories
Nathan Tobol, Codex Corporation and Chairman, IEEE 802 Local-Area
    Networking Subcommittee
Larry Nelson, AVCO Services
Barry Shein, Harvard University
Eliezer Gafni, MIT
Anat Gafni, Boston University
Josefina Bondoc, Boston University

Contacting Deitel & Associates

We would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

`deitel@deitel.com`

We will respond promptly.

Deitel Buzz® Online E-mail Newsletter

Our free e-mail newsletter, the *DEITEL® Buzz Online*, includes updates to *Operating Systems 3/e*, commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, errata, challenges, anecdotes, information on our corporate instructor-led training courses and more. To subscribe, visit

`www.deitel.com/newsletter/subscribe.html`

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in Internet/World Wide Web software technology, e-business/e-commerce software technology, object technology, operating systems and computer programming languages education. The company pro-

vides instructor-led courses on Internet and Web programming, wireless Internet programming, object technology, and major programming languages and platforms, including C, C++, Visual C++® .NET, Visual Basic® .NET, C#, Java, J2EE, XML, Perl, Python and more. The founders of Deitel & Associates, Inc., are Dr. Harvey M. Deitel and Paul J. Deitel. The company's clients include many of the world's largest computer companies, government agencies, branches of the military and business organizations. Through its 27-year publishing partnership with Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks, professional books, interactive CD-based multimedia *Cyber Classrooms*, *Complete Training Courses*, Web-based training courses and course management systems e-content for popular CMSs such as WebCT™, Blackboard™ and CourseCompass℠.

To learn more about Deitel & Associates, Inc., its publications and its corporate on-site training curriculum, see the last few pages of this book or visit:

    www.deitel.com

Individuals wishing to purchase Deitel® books, *Cyber Classrooms*, *Complete Training Courses* and Web-based training courses can do so through bookstores, online booksellers and the following Web sites:

    www.deitel.com
    www.prenhall.com/deitel
    www.InformIT.com/deitel
    www.InformIT.com/cyberclassrooms

Bulk orders by corporations and academic institutions should be placed directly with Prentice Hall. See www.prenhall.com/deitel for worldwide ordering instructions and to find your local Prentice Hall sales representative.

Welcome to the exciting world of operating systems. We sincerely hope you enjoy learning with this book.

Dr. Harvey M. Deitel
Paul J. Deitel
David R. Choffnes

About the Authors

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 42 years experience in the computing field, including extensive industry and academic experience. Dr. Deitel studied operating systems at the Massachusetts Institute of Technology where he earned B.S. and M.S. degrees. Dr. Deitel earned his Ph.D. from Boston University. While an undergraduate and master's candidate at MIT, he worked on the pioneering virtual-memory operating-systems projects at IBM (OS/360 and TSS/360) and MIT (Multics) that developed techniques now widely implemented in systems including UNIX, Linux and Windows

XP. He taught operating systems for 20 years in academia and industry. He earned tenure and served as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. He and Paul are the co-authors of several dozen books and multimedia packages and they are writing many more. With translations published in Japanese, German, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish, the Deitels' texts have earned international recognition. Dr. Deitel has delivered professional seminars to major corporations, government organizations and the military.

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of the Massachusetts Institute of Technology's Sloan School of Management, where he studied information technology and operating systems. Through Deitel & Associates, Inc., he has delivered hundreds of courses to Fortune 500, academic, government and military clients, and many other organizations. He has lectured for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world's best-selling computer science textbook authors.

David R. Choffnes is a graduate of Amherst College, where he earned degrees in Physics and French while pursuing advanced topics in Computer Science. His research interests include the fields of operating systems, computer architecture, computational biology and molecular computing. David has contributed to other Deitel publications including *Simply Java™ Programming* and *Simply Visual Basic® .NET.*