

---

# 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 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 or Java. The hardware topics required for an understanding 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, 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 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 commercial and open-source 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 Mac OS X, 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, as well as curriculum guidelines published by the IEEE

Computing Society and the Association for Computing Machinery (ACM). 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.

## 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, as well as operating systems for mobile devices. 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 is a discussion of threads, as well as an examination of issues related to multicore systems and parallel programming.
- **Memory management.** Chapters 8 and 9 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 management, and the effectiveness of a particular algorithm depends on the situation.
- **Storage management.** Chapters 10 through 13 describe how mass storage, the file system, 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. 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. 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 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.

- **Protection and security.** Chapters 14 and 15 discuss the mechanisms necessary for the protection and security of computer systems. The processes in an operating system must be protected from one another's activities, and 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 resources of the system. 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 16 and 17 discuss virtual machines and distributed systems. Chapter 16 is a new chapter that provides an overview of virtual machines and their relationship to contemporary operating systems. Included is an overview of the hardware and software techniques that make virtualization possible. Chapter 17 condenses and updates the three chapters on distributed computing from the previous edition. This change is meant to make it easier for instructors to cover the material in the limited time available during a semester and for students to gain an understanding of the core ideas of distributed computing more quickly.
- **Case studies.** Chapters 18 and 19 in the text, along with Appendices A and B (which are available on (<http://www.os-book.com>), present detailed case studies of real operating systems, including Linux, Windows 7, FreeBSD, and Mach. Coverage of both Linux and Windows 7 are presented throughout this text; however, the case studies provide much more detail. It is especially interesting to compare and contrast the design of these two very different systems. Chapter 20 briefly describes a few other influential operating systems.

## The Ninth Edition

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

1. Multicore systems
2. Mobile computing
3. Virtualization

To emphasize these topics, we have integrated relevant coverage throughout this new edition—and, in the case of virtualization, have written an entirely new chapter. 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.