

# Operating System Concepts

---

Second Edition

James L. Peterson  
Abraham Silberschatz

# Operating System Concepts

---

Second Edition

**James L. Peterson**  
**Abraham Silberschatz**

University of Texas at Austin



**Addison-Wesley Publishing Company**

Reading, Massachusetts • Menlo Park, California  
Don Mills, Ontario • Wokingham, England • Amsterdam  
Sydney • Singapore • Tokyo • Mexico City • Bogotá  
Santiago • San Juan

*To my parents, Wira and Mietek, my wife, Haya,  
and my children, Lemor, Sivan and Aaron.*

*Avi Silberschatz*

*To my wife, Jeanne,  
and my children, Jennifer and Kathryn.*

*Jim Peterson*

---

**Mark Dalton:** Sponsoring Editor

**Hugh Crawford:** Manufacturing Supervisor

**Karen Guardino:** Production Manager

**Thomas A. Philbrook and Barbara Atkinson:** Cover Designers

**Susan E. Vicenti:** Art Editor

**Natasha Wei:** Production Editor

This book is in the Addison-Wesley series in Computer Science.

**Michael A. Harrison:** Consulting Editor

**Library of Congress Cataloging in Publication Data**

Peterson, James Lyle.

Operating system concepts.

Bibliography: p.

Includes index.

1. Operating systems (Computers) I. Silberschatz,

Abraham. II. Title.

QA76.6.P475 1985 001.64'2 84-21637

ISBN 0-201-06198-8

Scope® Registered trademark of Control Data Corporation.

VMS™ Trademark of Digital Equipment Corporation

CP/M® Registered trademark of Digital Research Incorporated

UNIX™ Trademark of Bell Laboratories

Reproduced by Addison-Wesley from camera-ready copy prepared by the authors.

Copyright © 1985, 1983 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

# Preface

Operating systems are an essential part of a computer system. Similarly, a course on operating systems is an essential part of a computer science education. This book is intended as a text for an introductory course in operating systems at the junior, senior, or first-year graduate level. It provides a clear description of the concepts which underlie operating systems.

This book is not centered around any particular operating system or hardware. Instead, it discusses fundamental concepts which are applicable to a variety of systems. Our emphasis is on solving the problems encountered in designing an operating system, regardless of the underlying hardware on which the system will run.

## Content of this Book

The overall content of the book is as follows:

- 1 Introduction
- 2 Operating System Services
- 3 File Systems
- 4 CPU Scheduling
- 5 Memory Management
- 6 Virtual Memory
- 7 Disk and Drum Scheduling
- 8 Deadlocks
- 9 Concurrent Processes
- 10 Concurrent Programming
- 11 Protection
- 12 Design Principles
- 13 Distributed Systems
- 14 The Unix Operating System
- 15 Historical Perspective

As prerequisites, we assume the reader is familiar with general assembly language programming and computer organization. We do not discuss in any detail the characteristics of I/O devices or how to write device drivers.

Chapters 1, 2, and 3 explain what operating systems *are* and what they *do*. These chapters explain how the concept of an operating system has developed, the common features of an operating system, what it does for the user, and what it does for the computer system operator. It is motivational, historical, and explanatory in nature. We try to avoid how things are done internally in these chapters. Therefore, these chapters are suitable for individuals or lower-level classes who want to learn what an operating system is, without getting into the details of the internal algorithms.

Chapters 4 to 8 deal with the classic internal algorithms and structures of operating systems: *cpu scheduling*, *memory management*, and *device management*. They provide a firm practical understanding of the algorithms used: their properties, advantages, and disadvantages. The algorithms are presented in a natural order, so that new, more complicated systems can be built upon the understanding of simpler systems.

Chapter 9 introduces the unifying concept of the computer system as a collection of cooperating sequential processes. Chapters 10, 11, 12, and 13 present advanced topics and current trends, including high-level languages for writing concurrent programs, protection systems, design principles, and distributed systems. These topics are still being researched and may well need later revision. However, we include them in the book for two reasons. First, although research is still ongoing and final solutions to these problems are still being sought, there is general agreement that these topics are important and students should be exposed to them. Second, existing systems use these solutions, and anyone working with operating systems over the next five years will need to be aware of the developments in these directions.

In response to many requests, however, we have included a new chapter to illustrate how the many described concepts can be put together in a real system. We have chosen the Unix operating system, specifically Berkeley's 4.2BSD, for this example system. This operating system was chosen in part because it was at one time almost small enough to understand and yet is not a "toy" operating system. Most of its internal algorithms were selected for *simplicity*, not speed or sophistication. Unix is readily available to computer science departments, so many students may have used Unix.



Each chapter ends with references to further reading. Chapter 15 is essentially a set of references to further reading for the entire book, describing briefly some of the most influential operating systems.

## Organization

Operating systems first began to appear in the late 1950's, and for twenty years underwent major changes in concepts and technology. As a result, the first-generation operating system textbooks that appeared during this period (such as Brinch Hansen [1973a], Madnick and Donovan [1974], Shaw [1974], Tsichritzis and Bernstein [1974]) tried to explain a subject that changed even as they were being written.

Now, however, operating system theory and practice appears to have matured and stabilized. The fundamental operating system concepts are now well defined and well understood. While there will undoubtedly be new algorithms, the basic approach to cpu scheduling, memory management, the user interface, and so on, is not likely to change. Few really new operating systems are being written. Most large computers use operating systems that were designed in the 1960's. The newest operating systems are being developed for the multitude of microcomputer systems, but these are generally either CPM, Unix, or imitations of these. It is now possible to write a book that presents well-understood, agreed-upon, classic operating system material.

This text is one of a second generation of operating system textbooks (such as Calingaert [1982]). Our text differs from other texts in the level of content and organization. The basic concepts have been carefully organized and presented; the material flows naturally from these basic principles to more sophisticated ones.

The only controversial aspect of this book is its organization, specifically the definition of the formal process model as late as Chapter 9. Almost every other text places this material at the beginning as Chapter 2. In our experience, this arrangement does not work. The process model is a powerful and convenient unifying concept. However, when operating systems are first introduced, the student does not know the basic principles. To benefit from the process model, the student needs to understand how cpu scheduling and memory management can present an image of separate virtual processors, each with its own separate virtual memory space. Then, and only then, will the student really be able to understand why the process model is useful. Once the student has the proper background to be able to appreciate the process

model of operating systems, the standard material concerning processes, process coordination, synchronization, and communication can be presented.

Concurrency itself, in the form of overlapped I/O, spooling, multiprogramming, and time-sharing, is introduced as early as Chapter 1. However, we feel that the *formal* process model is best reserved until the basic concepts (cpu scheduling and memory management) are well understood.

## The Second Edition

Many comments and suggestions were forwarded to us concerning our first edition. These, together with our own observations while teaching at the University of Texas and IBM, have prodded us to produce this second edition. Our basic procedure was to reorganize and rewrite the material in each chapter, to bring some of the older material up-to-date, to improve the exercises, and to add a new chapter on Unix.

Substantive revisions were made in the following chapters:

- **Chapter 1.** A new organization clearly separates the performance and protection aspects of operating systems.
- **Chapter 3.** Sections have been reorganized to present a more natural flow of information, discussing first files, and then directories.
- **Chapter 8.** Additional examples illustrate the behavior of the various algorithms. Section 8.6, on recovery, has been rewritten to obtain a better organization.
- **Chapter 9.** Section 9.5 has been rewritten to bring the material up-to-date. In particular, a new, more concise definition of the critical section problem has been included, Dekker's algorithm for the synchronization of two processes has been replaced with Peterson's algorithm, and a new synchronization algorithm using special hardware instructions (that is, test and set) has been included. The section of the RC4000 system has been replaced by a discussion of the more modern Accent system.
- **Chapter 13.** A new subsection on the Byzantine general problem has been added.
- **Chapter 14.** A problem common to students (and professors) using the first edition was that they felt overwhelmed by the variety of solutions to the many aspects of operating systems. It was difficult

to see how a complete system would fit all the pieces together. To address this problem, we have added a chapter on a complete operating system. The initial problem was to create either a "paper" design (such as that used in Lister [1979]) or a real system. We decided to present a real system. The next problem was to choose a particular system. We chose Unix, specifically 4.2 BSD.

- **Chapter 15.** Since a new Chapter 14 was added, the old Chapter 14 (without the section on Unix) was renumbered Chapter 15.

## Errata

We have attempted to clean up every error in this book, but as with operating systems, there will undoubtedly still be some obscure bugs. We would appreciate it if you, the reader, would notify us of any errors or omissions in the book. If you would like to suggest improvements or contribute exercises, we would be glad to hear from you. An errata sheet is available to instructors for the first edition, and we will update it with errors in this edition as they become known.

## Acknowledgments

Eight years of CS 372 students at the University of Texas at Austin suffered through permutations of this material until we got it right. David Orshalick helped with the early table of contents. Dick Kiebert helped with the contents of Section 11.9 on language-based protection. During the writing stage, we were invited to design and teach an operating system course for IBM, which helped clarify our organization.

As the text was written, Carol Engelhardt deciphered our handwriting and edited our text into Scribe format. Carol's efforts throughout this project were the only thing that got it done.

Jeff Ullman helped us to get draft copies on the Dover at Stanford. Arthur Keller helped get those drafts back to Texas. Susan Lilly was able to understand what we were trying to say in the drafts and edit them into readable text. Elaine Rich, Richard Cohen, and Brian Reid explained the subtleties of Scribe, helping us to define our documents and make them work. The manuscript was read in various forms by Michael Molloy, Gael Buckley, and the reviewers.

During the lengthy revision process for this Second Edition, the Information Technology Center of Carnegie-Mellon University provided a supportive work environment.



Chapter 14 is derived from a draft by John Quarterman, who received comments on earlier drafts from Samuel J. Leffler, Bill Shannon, William N. Joy, and John B. Chambers. John Quarterman endured months of our questions and endless reviews as we struggled to write and rewrite the chapter. The credit for this chapter should go to John Quarterman, while any errors in presentation or fact for Chapter 14, as with the rest of the book, are, of course, ours.

Finally, we would like to acknowledge the helpful reviewing of Larry Flanigan, University of Michigan; William Appelbe, University of California at San Diego; Christopher Haynes, Indiana University; and Raymond Hookway, Case Western Reserve University.

J.P.  
A.S.

Alcatraz, Y. 368, 380  
Zero capacity 380, 395  
Zilles, S. M. 404, 587  
Zimmerman, H. 368  
Zucker, F. W. 455, 600

41973  
88.2.9

---

F8702/61 (英3-5/3880)  
操作系统概念 第2版

---

BG000680

# Contents

## Chapter 1 Introduction

1.1	What Is an Operating System?	1
1.2	Early Systems	4
1.3	Simple Monitor	6
1.4	Performance	9
1.5	Multiprogramming	17
1.6	Time Sharing	18
1.7	Real-Time Systems	21
1.8	Protection	22
1.9	Different Classes of Computers	31
1.10	Multiprocessor Systems	33
1.11	Summary	34
	Exercises	35
	Bibliographic Notes	37

## Chapter 2 Operating System Services

2.1	Types of Services	39
2.2	The User View	40
2.3	The Operating System View	49
2.4	Summary	54
	Bibliographic Notes	56

## Chapter 3 File Systems

3.1	File Concept	57
3.2	File Support	64
3.3	Access Methods	68
3.4	Allocation Methods	71
3.5	Directory Systems	82
3.6	File Protection	93
3.7	Implementation Issues	96
3.8	Summary	98
	Exercises	99
	Bibliographic Notes	102

## Chapter 4 CPU Scheduling

4.1	Review of Multiprogramming Concepts	103
4.2	Scheduling Concepts	105
4.3	Scheduling Algorithms	115
4.4	Algorithm Evaluation	129
4.5	Multiple Processor Scheduling	135
4.6	Summary	136
	Exercises	137
	Bibliographic Notes	141

## Chapter 5 Memory Management

5.1	Preliminaries	143
5.2	Bare Machine	145
5.3	Resident Monitor	146
5.4	Swapping	152
5.5	Multiple Partitions	156
5.6	Paging	172
5.7	Segmentation	183
5.8	Combined Systems	191
5.9	Summary	194
	Exercises	196
	Bibliographic Notes	200

## Chapter 6 Virtual Memory

6.1	Overlays	201
6.2	Demand Paging	204
6.3	Performance of Demand Paging	210
6.4	Page Replacement	213
6.5	Virtual Memory Concepts	216
6.6	Page Replacement Algorithms	217
6.7	Allocation Algorithms	228
6.8	Thrashing	232
6.9	Other Considerations	238
6.10	Summary	244
	Exercises	246
	Bibliographic Notes	254

**Chapter 7 Disk and Drum Scheduling**

7.1	Physical Characteristics	257
7.2	First-Come-First-Served Scheduling	261
7.3	Shortest-Seek-Time-First	262
7.4	SCAN	263
7.5	Selecting a Disk Scheduling Algorithm	265
7.6	Sector Queueing	266
7.7	Summary	268
	Exercises	268
	Bibliographic Notes	270

**Chapter 8 Deadlocks**

8.1	The Deadlock Problem	271
8.2	Deadlock Characterization	275
8.3	Deadlock Prevention	280
8.4	Deadlock Avoidance	283
8.5	Deadlock Detection	291
8.6	Recovery from Deadlock	295
8.7	Combined Approach to Deadlock Handling	298
8.8	Summary	300
	Exercises	301
	Bibliographic Notes	305

**Chapter 9 Concurrent Processes**

9.1	Precedence Graphs	307
9.2	Specification	310
9.3	Review of Process Concept	318
9.4	Hierarchy of Processes	320
9.5	The Critical Section Problem	323
9.6	Semaphores	340
9.7	Classical Process Coordination Problems	344
9.8	Interprocess Communication	349
9.9	Summary	361
	Exercises	361
	Bibliographic Notes	367



**Chapter 10 Concurrent Programming**

10.1	Motivation	369
10.2	Modularization	370
10.3	Synchronization	375
10.4	Concurrent Languages	393
10.5	Summary	400
	Exercises	401
	Bibliographic Notes	404

**Chapter 11 Protection**

11.1	Goals of Protection	407
11.2	Mechanisms and Policies	408
11.3	Domain of Protection	409
11.4	Access Matrix	410
11.5	Implementation of Access Matrix	411
11.6	Dynamic Protection Structures	415
11.7	Revocation	420
11.8	Existing Systems	422
11.9	Language-Based Protection	427
11.10	Protection Problems	432
11.11	Security	434
11.12	Summary	436
	Exercises	436
	Bibliographic Notes	438

**Chapter 12 Design Principles**

12.1	Goals	441
12.2	Mechanisms and Policies	442
12.3	Layered Approach	442
12.4	Virtual Machines	446
12.5	Multiprocessors	449
12.6	Implementation	450
12.7	System Generation	451
12.8	Summary	453
	Exercises	453
	Bibliographic Notes	455

## **Chapter 13 Distributed Systems**

13.1	Motivation	457
13.2	Topology	459
13.3	Communication	464
13.4	System Type	471
13.5	File Systems	474
13.6	Mode of Computation	476
13.7	Event Ordering	478
13.8	Synchronization	481
13.9	Deadlock Handling	486
13.10	Robustness	494
13.11	Reaching Agreement	496
13.12	Election Algorithms	499
13.13	Summary	502
	Exercises	503
	Bibliographic Notes	504

## **Chapter 14 The Unix Operating System**

14.1	History	507
14.2	Design Principles	510
14.3	Programmer Interface	512
14.4	User Interface	519
14.5	File System	523
14.6	Process Management	532
14.7	Memory Management	537
14.8	I/O System	541
14.9	Interprocess Communication	545
14.10	Summary	551
	Bibliographic Notes	552

## **Chapter 15 Historical Perspective**

15.1	Atlas	555
15.2	XDS-940	557
15.3	THE	557
15.4	RC 4000	558
15.5	CTSS	560
15.6	Multics	561
15.7	OS/360	561
15.8	Other Systems	563

# Chapter 13 Distributed Systems Bibliography 565

106	13.1 Motivation
	13.2 Topology
	13.3 Communication
	13.4 System Type
	13.5 File Systems
	13.6 Mode of Computation
	13.7 Event Ordering
	13.8 Synchronization
	13.9 Deadlock Handling
	13.10 Robustness
	13.11 Reaching Agreement
	13.12 Election Algorithms
	13.13 Summary
	Exercises
	Bibliographic Notes
504	
505	
507	
499	
496	
494	
486	
481	
478	
476	
471	
464	
460	
457	

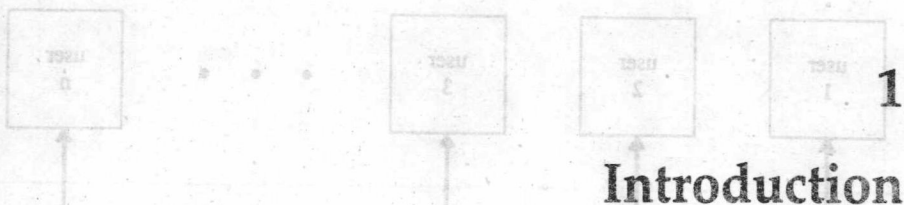
## Index

# Chapter 14 The Unix Operating System

525	14.1 History
	14.2 Design Principles
	14.3 Programmer Interface
	14.4 User Interface
	14.5 File System
	14.6 Process Management
	14.7 Memory Management
	14.8 IO System
	14.9 Interprocess Communication
	14.10 Summary
	Bibliographic Notes
522	
521	
545	
541	
537	
535	
523	
519	
512	
510	
507	

# Chapter 15 Historical Perspective

523	15.1 Atlas
	15.2 XDS-940
	15.3 THE
	15.4 RC 4000
	15.5 CTSS
	15.6 Multics
	15.7 OS/360
	15.8 Other Systems
561	
561	
560	
558	
557	
557	
555	



An *operating system* is a program which acts as an interface between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user may execute programs. The primary goal of an operating system is thus to make the computer system *convenient* to use. A secondary goal is to use the computer hardware in an *efficient* manner.

To understand what operating systems are, it is necessary to understand how they have developed. In this chapter, we trace the development of operating systems from the first hands-on systems to current multiprogrammed and time-shared systems. As we move through the various stages, we see how the components of operating systems evolved as natural solutions to problems in early computer systems. Understanding the reasons behind the development of operating systems gives an appreciation for what an operating system does and how it does it.

## 1.1 What Is an Operating System?

An operating system is an important part of almost every computer system. A computer system can be roughly divided into 4 components (Figure 1.1):

- The **hardware** (cpu, memory, I/O devices).
- The **operating system**.
- The **applications programs** (compilers, database systems, video games, business programs).
- The **users** (people, machines or other computers).

The hardware provides the basic computing resources. The applications programs define the ways in which these resources are used to solve the