

Operating Systems Principles

Lubomir F. Bic

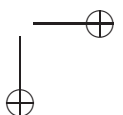
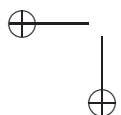
University of California, Irvine

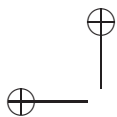
Alan C. Shaw

University of Washington, Seattle

Pearson
Education

PEARSON EDUCATION INC. *Upper Saddle River, New Jersey 07458*





Library of Congress Cataloging-in-Publication Data

Vice President and Editorial Director, ECS:
Publisher:
Associate Editor:
Editorial Assistant:
Vice President and Director of Production and Manufacturing, ESM:
Executive Managing Editor:
Assistant Managing Editor:
Production Editor:
Director of Creative Services:
Creative Director:
Art director:
Cover Designer:
Art Editor:
Manufacturing Manager:
Manufacturing Buyer:
Marketing Manager:
Marketing Assistant:



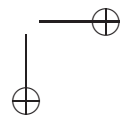
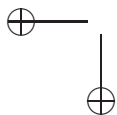
© 2003 by Pearson Education, Inc.
Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

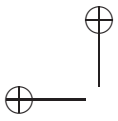
*All rights reserved. No part of this book may
be reproduced, in any form or by any means,
without permission in writing from the publisher.*

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN

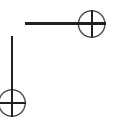
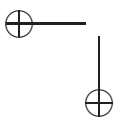
Pearson Education LTD., *London*
Pearson Education Australia PTY, Limited, *Sydney*
Pearson Education Singapore, Pte. Ltd
Pearson Education North Asia Ltd, *Hong Kong*
Pearson Education Canada, Ltd., *Toronto*
Pearson Educación de México, S.A. de C.V.
Pearson Education - Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd

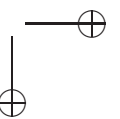
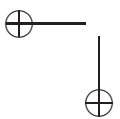
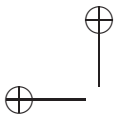




*To Zuzana and Alexander
Lubomir Bic*

*To Elizabeth
Alan Shaw*







Preface

Operating systems bridge the gap between the hardware of a computer system and the user. Consequently, they are strongly influenced by hardware technology and architecture, both of which have advanced at a breathtaking pace since the first computers emerged in the 1940s. Many changes have been quantitative: the speed of processors, memories, and devices has been increasing continuously, whereas their size, cost, and power consumption has been decreasing. But many qualitative changes also have occurred. For example, personal computers with sophisticated input, output, and storage devices are now omnipresent; most also are connected to local area networks or the Internet. These advances have dramatically reshaped the world within which operating systems must exist and cooperate. Instead of managing a single processor controlling a collection of local memories and I/O devices, contemporary operating systems are required to manage highly parallel, distributed, and increasingly more heterogeneous configurations.

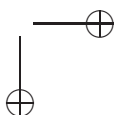
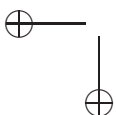
This book is an introduction to operating systems, appropriate for computer science or computer engineering majors at the junior or senior level. One objective is to respond to a major paradigm shift from single-processor to distributed and parallel computer systems, especially in a world where it is no longer possible to draw a clear line between operating systems for centralized environments and those for distributed ones. Although most of the book is devoted to traditional topics, we extend and integrate these with basic ideas in distributed computing.

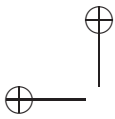
CONTENTS

After the introductory chapter, the book is organized into four main sections: Process Management and Coordination, Memory Management, File and I/O Management, and Protection and Security. At the end of each chapter, there is a list of the key concepts, terms, and abbreviations defined in the chapter; the back of the book contains a glossary.

Processes and Threads

Processes and, more recently, threads, are the basis of concurrency and parallelism, and have always been prominent parts of the study of operating systems. This area can be subdivided into two components: the creation of processes or threads, and their coordination. In Chapters 2 and 3, we treat the topic from the programming point of view, presenting a spectrum of constructs for expressing concurrency and for coordinating the execution of the resulting processes or threads. This includes the coordination of processes in a distributed environment, which must be based ultimately on message-passing rather than shared variables. In Chapters 4 and 5, we examine the problem from the implementation point of view by presenting the necessary data structures and operations to implement and manage processes and threads at the operating systems level. This discussion also includes issues of process and threads scheduling, interrupt handling, and other kernel functions. Chapter 6 is concerned with the important problem of deadlocks in both centralized and distributed systems.





vi Preface

Main Memory

Main memory has always been a scarce resource, and much of the past operating systems research has been devoted to its efficient use. Many of these results have become classical topics of operating systems; these are covered in Chapters 7, 8, and 9. Among these topics are techniques for physical memory allocation, implementation of virtual memory using paging or segmentation, and static and dynamic sharing of data and code. We also present the principles of distributed shared memory, which may be viewed as an extension of virtual memory over multiple computers interconnected by a communication network.

File Systems and I/O

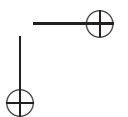
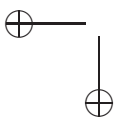
Files were devised in the early days of computing as a convenient way to organize and store data on secondary storage devices. Although the devices have evolved dramatically, the basic principles of files have not. In Chapter 10, we discuss file types and their representations on disks or tapes. We also present ways of organizing and implementing file directories. In recent years, the most significant developments in the file systems area have been driven by the proliferation of networking. Many systems today do not maintain their own file systems on local drives. Instead, a more typical configuration is a network of machines, all accessing dedicated file servers. Frequently, the file systems are distributed over multiple servers or multiple networks. The last section of the chapter addresses file systems issues in such distributed environments.

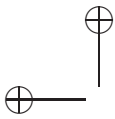
Hiding the details of individual I/O devices by supporting higher-level abstractions has always been one of the main tasks of operating systems. Modern systems must continue to provide this essential service, but with a larger variety of faster and more sophisticated devices. Chapter 11 is devoted to this topic, presenting the principles of polling, interrupts, and DMA, as employed by various device drivers. Also discussed are device-independent aspects of I/O processing, including buffering and caching, error-handling, and device scheduling.

Protection and Security

Protecting a computing facility from various attacks requires a broad spectrum of safeguards. Chapter 12 focuses on the protection and security interface of the system, which guards the system access. This requires authentication of users, remote services, and clients. Despite many technological breakthroughs, user authentication still relies largely on passwords presented by users at the time of login. But the existence of computer networks has again stimulated the most dramatic developments in protection and security: the vulnerability of communication lines makes it necessary to employ techniques in secret or public key cryptography. We discuss the application of cryptographic methods both to protect information transmitted between computers and to verify its authenticity.

Once a user has entered the system, the system must control the set of resources accessible to that user. This is accomplished by hardware mechanisms at the instruction level and by access or capability lists at the software level. In addition, mechanisms to prevent unauthorized flow of information among different users also must be provided. Chapter 13 discusses internal protection mechanisms.





EXERCISES AND PROGRAMMING PROJECTS

Each chapter ends with a set of exercises reflecting the presented topics. The exercises have been chosen carefully to satisfy the needs of different teaching styles. Each exercise set contains both analytical and constructive exercises, where students must apply conceptual knowledge acquired from the chapter to solve specific problems. We also have included questions that lend themselves to discussion or speculative analysis. A solutions manual is available to professors; they can obtain a copy from their local Prentice-Hall representative.

The set of five large programming projects and several smaller programming exercises at the end of the book are designed to complement the conceptual understanding gained from the book with practical hands-on experience. They may be used selectively as term projects or can serve as the basis for a separate laboratory component in operating systems.

APPROACH AND PHILOSOPHY

As expected, we provide in-depth coverage of all standard topics in the field of operating systems. A conventional approach typically also includes separate chapters on operating systems support for distributed network-based environments, usually appearing at the end of the text. The problem with this organization is that it makes an artificial distinction between centralized and distributed systems. In reality, there is often no clear demarcation line between the two, and they have many issues in common. Concurrency and parallelism have always been a major topic of operating systems. Even the earliest mainframes of the 1950s and 1960s attempted to overlap CPU execution with I/O processing to achieve better utilization of both. Advanced programming techniques of the 1970s and 1980s made it necessary to support concurrent processes at the user level, leading operating systems designers to provide new process synchronization and scheduling techniques, many of which also apply to networked environments. The last two decades have forced software manufacturers to seriously consider networking and physical distribution, and to integrate the necessary tools and techniques into their operating systems products.

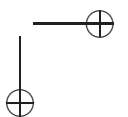
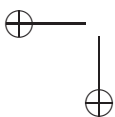
We have chosen to preserve the natural relationship and overlap between centralized and distributed operating systems issues by integrating them within each chapter. The main distributed operating systems topics presented include message-based synchronization and remote procedure calls, distributed deadlocks, distributed shared memory, distributed file systems, and secure communication using cryptography.

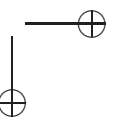
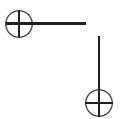
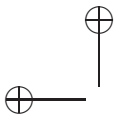
Following the above philosophy, we also have refrained from presenting case studies of existing operating systems in separate chapters. Instead, we have distributed and integrated all case studies—from Unix, Linux, Windows, and many other influential operating systems—throughout the chapters. They illustrate the relevance of each concept at the time of its presentation.

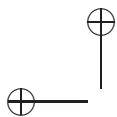
May 20002

Lubomir Bic

Alan Shaw

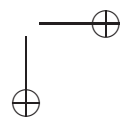
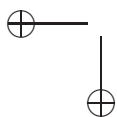


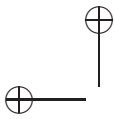




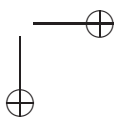
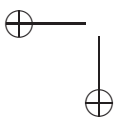
Contents

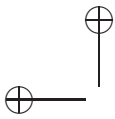
1	Introduction	1
1.1	The Role of Operating Systems	1
1.1.1	Bridging the Hardware/Application Gap	1
1.1.2	Three Views of Operating Systems	6
1.2	Organization of Operating Systems	11
1.2.1	Structural Organization	11
1.2.2	The Hardware Interface	12
1.2.3	The Programming Interface	15
1.2.4	The User Interface	17
1.2.5	Runtime Organization	24
1.3	Operating System Evolution and Concepts	25
1.3.1	Early Systems	26
1.3.2	Batch Operating Systems	27
1.3.3	Multiprogramming Systems	28
1.3.4	Interactive Operating Systems	30
1.3.5	Personal Computer and Workstation Operating Systems	31
1.3.6	Real-Time Operating Systems	32
1.3.7	Distributed Operating Systems	33
	Part One Process Management and Coordination	37
2	Basic Concepts: Processes and Their Interactions	39
2.1	The Process Notion	39
2.2	Defining and Instantiating Processes	41
2.2.1	Precedence Relations Among Processes	41
2.2.2	Implicit Process Creation	44
2.2.3	Explicit Process Creation with <i>fork</i> and <i>join</i>	47
2.2.4	Process Declarations and Classes	51
2.3	Basic Process Interactions	52
2.3.1	Competition: The Critical Section Problem	52
2.3.2	Cooperation	58
2.4	Semaphores	59
2.4.1	Semaphore Operations and Data	60
2.4.2	Mutual Exclusion with Semaphores	61
2.4.3	Semaphores in Producer/Consumer Situations	62
2.5	Event Synchronization	64





x	Contents	
3	Higher-Level Synchronization and Communication	70
3.1	Shared Memory Methods	71
3.1.1	Monitors	71
3.1.2	Protected Types	76
3.2	Distributed Synchronization and Communication	77
3.2.1	Message-Based Communication	77
3.2.2	Procedure-Based Communication	83
3.2.3	Distributed Mutual Exclusion	87
3.3	Other Classic Synchronization Problems	90
3.3.1	The Readers/Writers Problem	90
3.3.2	The Dining Philosophers Problem	92
3.3.3	The Elevator Algorithm	94
3.3.4	Event Ordering with Logical Clocks	97
4	The Operating System Kernel: Implementing Processes and Threads	105
4.1	Kernel Definitions and Objects	105
4.2	Queue Structures	108
4.2.1	Resource Queues in an Operating System	108
4.2.2	Implementations of Queues	109
4.3	Threads	112
4.4	Implementing Processes and Threads	114
4.4.1	Process and Thread Descriptors	114
4.4.2	Implementing Operations on Processes	120
4.4.3	Operations on Threads	123
4.5	Implementing Synchronization and Communication Mechanisms	123
4.5.1	Semaphores and Locks	124
4.5.2	Monitor Primitives	128
4.5.3	Clock and Time Management	130
4.5.4	Communication Primitives	136
4.6	Interrupt Handling	139
5	Process and Thread Scheduling	147
5.1	Organization of Schedulers	147
5.1.1	Embedded and Autonomous Schedulers	147
5.1.2	Priority Scheduling	149
5.2	Scheduling Methods	151
5.2.1	A Framework for Scheduling	151
5.2.2	Common Scheduling Algorithms	154
5.2.3	Comparison of Methods	159
5.3	Priority Inversion	168
5.4	Multiprocessor and Distributed Scheduling	170
6	Deadlocks	177
6.1	Deadlock with Reusable and Consumable Resources	178
6.1.1	Reusable and Consumable Resources	178
6.1.2	Deadlocks in Computer Systems	179
6.2	Approaches to the Deadlock Problem	181





6.3	A System Model	182
6.3.1	Resource Graphs	182
6.3.2	State Transitions	183
6.3.3	Deadlock States and Safe States	184
6.4	Deadlock Detection	186
6.4.1	Reduction of Resource Graphs	187
6.4.2	Special Cases of Deadlock Detection	187
6.4.3	Deadlock Detection in Distributed Systems	189
6.5	Recovery from Deadlock	192
6.5.1	Process Termination	192
6.5.2	Resource Preemption	193
6.6	Dynamic Deadlock Avoidance	194
6.6.1	Claim Graphs	194
6.6.2	The Banker’s Algorithm	194
6.7	Deadlock Prevention	198
6.7.1	Eliminating the Mutual-Exclusion Condition	198
6.7.2	Eliminating the Hold-and-Wait Condition	198
6.7.3	Eliminating the Circular-Wait Condition	199

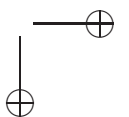
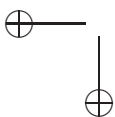
Part Two Memory Management 205

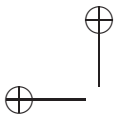
7 Physical Memory 207

7.1	Preparing a Program for Execution	207
7.1.1	Program Transformations	207
7.1.2	Logical-to-Physical Address Binding	208
7.2	Memory Partitioning Schemes	212
7.2.1	Fixed Partitions	213
7.2.2	Variable Partitions	214
7.2.3	The Buddy System	218
7.3	Allocation Strategies for Variable Partitions	220
7.3.1	Measures of Memory Utilization	221
7.4	Managing Insufficient Memory	224
7.4.1	Memory Compaction	224

8 Virtual Memory 231

8.1	Principles of Virtual Memory	231
8.2	Implementations of Virtual Memory	233
8.2.1	Paging	233
8.2.2	Segmentation	240
8.2.3	Paging with Segmentation	241
8.2.4	Paging of System Tables	242
8.2.5	Translation Look-Aside Buffers	245
8.3	Memory Allocation in Paged Systems	246
8.3.1	Global Page Replacement Algorithms	249
8.3.2	Local Page Replacement Algorithms	256
8.3.3	Load Control and Thrashing	262
8.3.4	Evaluation of Paging	266



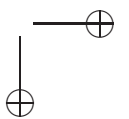
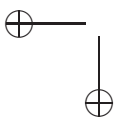


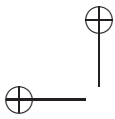
xii Contents

9	Sharing of Data and Code in Main Memory	274
9.1	Single-Copy Sharing	274
9.1.1	Reasons for Sharing	274
9.1.2	Requirements for Sharing	275
9.1.3	Linking and Sharing	277
9.2	Sharing in Systems without Virtual Memory	278
9.3	Sharing in Paging Systems	279
9.3.1	Sharing of Data	279
9.3.2	Sharing of Code	281
9.4	Sharing in Segmented Systems	283
9.4.1	Sharing of Code and Data	283
9.4.2	Unrestricted Dynamic Linking	284
9.5	Principles of Distributed Shared Memory	287
9.5.1	The User’s View of Distributed Shared Memory	288
9.6	Implementations of Distributed Shared Memory	290
9.6.1	Implementing Unstructured Distributed Shared Memory	290
9.6.2	Implementing Structured Distributed Shared Memory	296

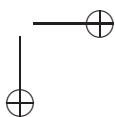
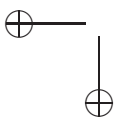
Part Three File Systems and Input/Output 303

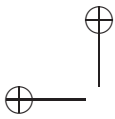
10	File Systems	305
10.1	Basic Functions of File Management	305
10.2	Hierarchical Model of a File System	306
10.3	The User’s View of Files	309
10.3.1	File Names and Types	309
10.3.2	Logical File Organization	311
10.3.3	Other File Attributes	313
10.3.4	Operations on Files	314
10.4	File Directories	315
10.4.1	Hierarchical Directory Organizations	316
10.4.2	Operations on Directories	322
10.4.3	Implementation of File Directories	325
10.5	Basic File System	329
10.5.1	File Descriptors	329
10.5.2	Opening and Closing Files	330
10.6	Device Organization Methods	333
10.6.1	Contiguous Organization	334
10.6.2	Linked Organization	335
10.6.3	Indexed Organization	336
10.6.4	Management of Free Storage Space	337
10.7	Principles of Distributed File Systems	339
10.7.1	Directory Structures and Sharing	339
10.7.2	Semantics of File Sharing	343
10.8	Implementing Distributed File System	344
10.8.1	Basic Architecture	344
10.8.2	Caching	345





	Contents	xiii
10.8.3 Stateless Versus Stateful Servers		346
10.8.4 File Replication		349
11 Input/Output Systems		357
11.1 Basic Issues in Device Management		357
11.2 A Hierarchical Model of the Input/Output System		359
11.2.1 The Input/Output System Interface		359
11.3 Input/Output Devices		363
11.3.1 User Terminals		363
11.3.2 Printers and Scanners		366
11.3.3 Secondary Storage Devices		367
11.3.4 Performance Characteristics of Disks		370
11.3.5 Networks		372
11.4 Device Drivers		373
11.4.1 Memory-Mapped Versus Explicit Device Interfaces		375
11.4.2 Programmed Input/Output with Polling		376
11.4.3 Programmed Input/Output with Interrupts		379
11.4.4 Direct Memory Access		383
11.5 Device Management		386
11.5.1 Buffering and Caching		386
11.5.2 Error Handling		392
11.5.3 Disk Scheduling		397
11.5.4 Device Sharing		400
 Part Four Protection and Security		 405
12 The Protection and Security Interface		407
12.1 Security Threats		407
12.1.1 Damage Types		408
12.1.2 Vulnerable Resources		409
12.1.3 Attack Types		410
12.2 Functions of a Protection System		418
12.2.1 External Safeguards		418
12.2.2 Verification of User Identity		419
12.2.3 Communication Safeguards		420
12.2.4 Threat Monitoring		420
12.3 User Authentication		420
12.3.1 Approaches to Authentication		420
12.3.2 Passwords		422
12.4 Secure Communication		426
12.4.1 Principles of Cryptography		426
12.4.2 Secret-Key Cryptosystems		428
12.4.3 Public-Key Cryptosystems		433
 13 Internal Protection Mechanisms		 442
13.1 The Access Control Environment		442





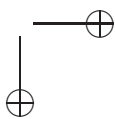
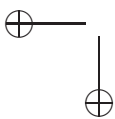
xiv Contents

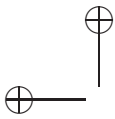
13.2	Instruction-Level Access Control	443
13.2.1	Register and Input/output Protection	443
13.2.2	Main Memory Protection	444
13.3	High-Level Access Control	450
13.3.1	The Access Matrix Model	450
13.3.2	Access Lists and Capability Lists	452
13.3.3	A Comprehensive Example: Client/Server Protection	461
13.3.4	Combining Access Lists and Capability Lists	463
13.4	Information Flow Control	464
13.4.1	The Confinement Problem	464
13.4.2	Hierarchical Information Flow	467
13.4.3	The Selective Confinement Problem	469

Part Five Programming Projects 475

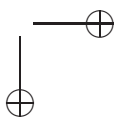
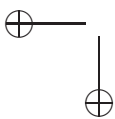
PROJECTS

I	Process/Thread Synchronization	477
1	Project Overview	477
2	Setting Up a Race Condition	477
3	Solutions to the Critical Section Problem	478
3.1	Solution Using <i>mutex</i> Locks	478
3.2	Software Solution	479
4	Implementing General Semaphores	479
4.1	Solution Using <i>Mutex</i> Locks and <i>Condition</i> Variables	479
4.2	Software Solution	479
5	Bounded Buffer	480
6	Summary of Specific Tasks	480
7	Ideas for Additional Tasks	480
II	Process and Resource Management	482
1	Project Overview	482
2	Basic Process and Resource Manager	482
2.1	Process States	482
2.2	Representation of Processes	483
2.3	Representation of Resources	483
2.4	Operations on Processes and Resources	484
2.5	The Scheduler	485
2.6	The Presentation Shell	487
3	Extended Process and Resource Manager	488
3.1	Timeout Interrupts	488
3.2	Input/Output Processing	489
3.3	The Extended Shell	489
4	Summary of Specific Tasks	490
5	Ideas for Additional Tasks	490





III Main Memory Management	492
1 Project Overview	492
2 The Memory Manager	492
2.1 Main Memory	492
2.2 The User Interface	493
3 The Simulation Experiment	493
3.1 Generating Request Sizes	494
3.2 Gathering Performance Data	495
3.3 Choosing a Block to Release	495
4 Summary of Specific Tasks	495
5 Ideas for Additional Tasks	495
IV Page Replacement Algorithms	496
1 Project Overview	496
2 Global Page Replacement Algorithms	496
3 Local Page Replacement Algorithms	497
4 Generating Reference Strings	498
5 Performance Evaluations	499
6 Summary of Specific Tasks	500
7 Ideas for Additional Tasks	500
V File System	501
1 Project Overview	501
2 The Input/Output System	501
3 The File System	502
3.1 Interface Between User and File System	502
3.2 Organization of the File System	502
3.3 The Directory	503
3.4 Creating and Destroying a File	503
3.5 Opening and Closing a File	504
3.6 Reading, Writing and Seeking in a File	504
3.7 Listing the Directory	505
4 The Presentation Shell	505
5 Summary of Specific Tasks	506
6 Ideas for Additional Tasks	506
Other Programming Projects	507
1 Timer Facility	507
2 Process Scheduling	507
3 The Banker’s Algorithm	508
4 Disk Scheduling Algorithm	508
5 Stable Storage	509
Glossary	510
Bibliography	525
Index	529



Operating Systems Principles & Practice Volume IV: Persistent Storage Second Edition Thomas Anderson University of Washington Mike Dahlin University of Texas and Google Recursive Books recursivebooks.com. Operating Systems: Principles and Practice (Second Edition) Volume IV: Persistent Storage by Thomas Anderson and Michael Dahlin Copyright ©Thomas Anderson and Michael Dahlin, 2011-2015. ISBN 978-0-9856735-6-7 Publisher: Recursive Books, Ltd., <http://recursivebooks.com/> Cover: Reflection Lake, Mt. Operating Systems: Principles and Practice Tom Anderson How This Course Fits in the UW CSE Curriculum • CSE 333: Systems Programming • Project experience in C/C++ • How to use the operating system interface • CSE 451: Operating Systems • How to make a single computer work reliably • How an operating system works internally • CSE 452: Distributed Systems (winter 2015) • How to make a set of computers. • Question • How should an operating system allocate processing time between competing uses? Operating Systems: Internals and Design Principles. Read more. Formal Refinement of Operating System Kernels. Read more. The design of the UNIX operating system. Read more. Solaris Operating Environment. System Administrator's Guide. Read more. • This page intentionally left blank Operating System Concepts Essentials This page intentionally left blank Opera Operating system concepts. Microsoft Windows Operating System Essentials. —. Report "Operating System Principles". Your name. Email.

Operating Systems: Intern has been added to your Cart. Add to Cart. Buy Now. See and discover other items: operating system, operating systems, principles of design, Operating System Softwares. There's a problem loading this menu right now. Learn more about Amazon Prime.