



OPERATING SYSTEMS

A Design-Oriented Approach

Charles Crowley

OPERATING SYSTEMS

A Design-Oriented Approach

Charles Crowley
University of New Mexico

IRWIN

Chicago • Bogota • Boston • Buenos Aires • Caracas
London • Madrid • Mexico City • Sydney • Toronto



IRWIN Concerned about Our Environment

In recognition of the fact that our company is a large end-user of fragile yet replenishable resources, we at IRWIN can assure you that every effort is made to meet or exceed Environmental Protection Agency (EPA) recommendations and requirements for a "greener" workplace.

To preserve these natural assets, a number of environmental policies, both companywide and department-specific, have been implemented. From the use of 50% recycled paper in our textbooks to the printing of promotional materials with recycled stock and soy inks to our office paper recycling program, we are committed to reducing waste and replacing environmentally unsafe products with safer alternatives.

© Richard D. Irwin, a Times Mirror Higher Education Group, Inc., company, 1997

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.

Irwin Book Team

Publisher: *Tom Casson*

Senior sponsoring editor: *Elizabeth A. Jones*

Senior developmental editor: *Kelley Butcher*

Project supervisor: *Lynne Basler*

Senior Production supervisor: *Laurie Sander*

Director, Prepress Purchasing: *Kimberly Meriwether David*

Compositor: *Interactive Composition Corporation*

Typeface: *10/12 Times Roman*

Printer: *Times Mirror Higher Education Group, Inc., Print Group*



Times Mirror

Higher Education Group

Library of Congress Cataloging-in-Publication Data

Crowley, Charles (Charles Patrick)

Operating systems : a design-oriented approach / Charles Crowley.

p. cm.

Includes bibliographical references and index.

ISBN 0-256-15151-2

1. Operating systems (Computers) I. Title.

QA76.76.063C77 1997

005.4'3—dc20

96-43708

Printed in the United States of America

2 3 4 5 6 7 8 9 0 WCB 3 2 1 0 9 8 7

PREFACE

In this book I have tried to approach the traditional, junior or senior level operating systems course in a new way. There are several areas where I have done things differently:

- *Describe the external operating system interface:* I start with a description of the system call interface to an operating system.
- *Use of code:* I have tried to steer a middle course between a concepts approach and a case study approach.
- *Development of concepts:* I have tried to show how the operating system concepts developed into their present form.
- *Design orientation:* I have tried to show how ideas from the design of operating systems relate to the design of other types of programs.

EXTERNAL OPERATING SYSTEM INTERFACE

Many students come to an operating system class without a clear understanding of what an operating system really does. These students should understand how operating system services are used before learning how these services are implemented. To deal with this, the book begins with a simplified, UNIX-like set of system calls. The book includes a discussion of these calls, example programs using these calls, and a simple shell program to integrate the examples.

CONCEPTS OR CASE STUDIES

There have always been two approaches to the operating systems class. The first approach is the *concept* or *theory* approach which concentrates on the basic conceptual issues in the design of operating systems. These courses discuss each of the basic problems in operating systems design and the range of common solutions to those problems. The books are mostly text and diagrams with very little code. The second approach is the *case study* method which concentrates on an example operating system that is simple but complete. The books contain a lot of code and spend a lot of pages explaining the code in detail.

There are advantages and disadvantages to both approaches. Some people feel the ideal situation is to take both classes but this is rarely possible in an already crowded computer science curriculum so one is required to make a choice.

USE OF CODE

I have tried to find a middle course between the two approaches. This book is basically a concepts oriented book with more code than is usual. I have found that seeing actual code allows the students to understand the concepts more deeply, feel more comfortable about

the material, and ask questions they wouldn't have thought to ask in a purely concepts oriented course. The code does not comprise a complete operating system however and it as simple as possible in order to reduce the amount of pages devoted to explaining it.

DEVELOPMENT OF CONCEPTS

I have tried to show how these ideas developed. Many of the concepts in operating systems have developed over many years and the current solutions were developed slowly, in several stages. Each new solution had a problem that the next solution tried to fix. I think it helps the student to understand this development and see that these ideas were not brilliant flashes of insight that came out of nowhere but ideas that were improved by many people over many years. The development was a series of good ideas where each improvement made sense in the context in which it was developed. Seeing this development helps to understand why the solutions have their present form. In addition, it is useful to know the design constraints that caused solutions to develop into their present form because technological advances often change these constraints and old solutions that used to be inferior suddenly become practical again. Finally these developments give students examples of the design process through a series of potential but flawed solutions to a problem to a final solution that is acceptable.

DESIGN ORIENTATION

Finally there is a concentration on design. In some ways designing an operating system is a pretty specialized activity having to do mainly with resource management. But many basic design ideas run through all designs and they show up in operating systems as much as anywhere else. Throughout the book I note places where we are presented with typical design problems. I abstract the operating systems related problems and solutions from the book into the general design problems and solutions and present them in a way that they can be applied to design problems in other areas of computer science.

Clearly it is not possible to cover all design topics and issues. I am striving for two things. First, I want to give the student an awareness of design issues, where they come up, which techniques to apply, how they can be generalized, etc. I do not present an organized survey of design techniques but a series of useful ones that come up in the context of operating systems. I hope to make the student aware of design and to enable the student to start doing their own generalizing about design. Second, I present a collection of useful design techniques that the students can use in their design toolkit.

Very few computer professionals will participate in the design of an operating system during the course of their careers. While it is important that students of computer science have a good foundation in the basic concepts in specialized areas such as operating systems, it is not necessary that every computer science student

understand all the details. However, there is a thread running through all areas of software engineering: the concept of design. There are many issues which are tackled during the design of an operating system which can be generalized and applied to other areas of computer science. In this book, I attempt to focus on these design issues and their implications for other areas.

I have oriented this book to provide a solid preparation for the larger design projects the student will encounter in later software engineering courses and as preparation for their career as a software professional designing, implementing and maintaining a wide variety of systems. This orientation also enables the operating systems course follows modern developments in the field of computer science. The interrelations between the separate areas of computer science are becoming more important. For example, in the area of high speed parallel machines, it is clear that it is necessary to think of the hardware, the operating system, and the programming language as a single system to get maximum performance. Optimization in any part of the system will have consequences for the other parts of the system.

The design techniques are noted in side bars as they come up and longer explanations of each design topic are placed in separate chapters from the operating system material. The instructor can structure a course with varying degrees of concentration on design aspects. The goal is that the design sections are independent of the main flow of the text and independent from each other. This will allow the instructor to pick and choose those design sections he or she finds to be useful.

USING THIS BOOK IN A COURSE

There is more material in this book than can be comfortably covered in a one-semester course. A number of the sections of the book have been marked with an asterisk. This indicates that they can be skipped with no loss of continuity in the presentation. In addition, all of the design chapters and design sidebars can be skipped with no loss of continuity. If you skip all the design chapters you can probably just cover the book in a semester. I expect that most instructors will choose to skip some of the design sections and some of the optional sections and teach a course that is about 90 percent operating systems and 10 percent design issues. Alternatively, the design issues could be covered in a separate one-unit course, strictly on design, that is taken along with the operating systems course.

ACKNOWLEDGMENTS

I want to thank my colleague Barney Maccabe who started out as my co-author on the book but had too many other commitments to work on the book. Our discussions lead to the conception of the book and the design orientation. I have talked with Barney on many aspects of operating systems and computer science and those discussions shaped many of my ideas.

I also want to thank John Brayer who used drafts of the book in several of his operating systems courses and put up with many typos, badly written sections, and unfinished sections. His comments helped in the development of the book.

I also want to thank my other colleagues in the Computer Science Department at the University of New Mexico. Continuing “in-the-hall” discussions with them have helped me formulate and improve my ideas.

Many students in CS 481 have used drafts of the books and have provided many useful comments and suggestions. I want to particularly thank Dave Rosenbaum who provided extensive comments on the entire draft, found numerous errors and typos and taught me some things about writing good English as well.

My treatment of the design chapters was greatly influenced by the design patterns developments in recent years and particularly the fine book by Gamma et al. (1995). I used their presentation format in the design chapters.

I want to thank all the people who reviewed this book while it was being developed. I corrected many problems and got many ideas for the presentation from their careful reviews and comments. The design chapters especially profited from their comments.

- Mustaque Ahamad, Georgia Institute of Technology
- Jim Alves-Foss, University of Idaho
- Anish Arora, The Ohio State University
- Brent Auernheimer, California State University - Fresno
- Anthony Q. Baxter, University of Kentucky
- Mahesh Dodani, University of Iowa
- H. George Friedman, Jr., University of Illinois - Urbana
- Tim Gottleber
- Stephen J. Hartley, Drexel University
- Giorgio P. Ingargiola, Temple University
- Stephen J. Krebsbach, South Dakota State University
- Donald S. Miller, Arizona State University
- Matt W. Mutka, Michigan State University
- Richard Newman-Wolfe, University of Florida
- Steve Reichenbach, University of Nebraska - Lincoln
- Bernhard Weinberg, Michigan State University

Finally my wife, Ella Sitkin, put up with many bad moods and excuses that I was “too busy” during the years this book was developed. As if that wasn’t enough, she also edited several chapters.

CONTENTS

1

Introduction 1

- 1.1 Where Does an Operating System Fit in? 2
 - 1.1.1 System Levels 2
- 1.2 What Does an Operating System Do? 4
 - 1.2.1 Hardware Resources 4
 - 1.2.2 Resource Management 5
 - 1.2.3 Virtual Computers 6
- 1.3 A Virtual Computer 8
 - 1.3.1 Virtual Processor 8
 - 1.3.2 Virtual Primary Memory 10
 - 1.3.3 Virtual Secondary Memory 10
 - 1.3.4 Virtual I/O 10
- 1.4 Do We Need an Operating System? 11
- 1.5 Summary 12
 - 1.5.1 Terminology 13
 - 1.5.2 Review Questions 14
 - 1.5.3 Further Reading 14
- 1.6 Problems 14

2

The Hardware Interface 16

- 2.1 The CPU 17
 - 2.1.1 General-Purpose Registers 17
 - 2.1.2 Control Registers 17
 - 2.1.3 Processor Modes 18
 - 2.1.4 Instruction Set 18
 - 2.1.5 Machine Instructions in C++ Code 19
- 2.2 Memory and Addressing 19
- 2.3 Interrupts 20
- 2.4 I/O Devices 21
 - 2.4.1 Disk Controller 22
- 2.5 Summary 23
 - 2.5.1 Terminology 23
 - 2.5.2 Review Questions 23
 - 2.5.3 Further Reading 24
- 2.6 Problems 24

3

The Operating System Interface 25

- 3.1 What Are System Calls? 26
 - 3.1.1 How to Make a System Call 27
 - 3.1.2 What Is a System Call Interface? 28
- 3.2 An Example System Call Interface 28
 - 3.2.1 System Call Overview 28
 - 3.2.2 Hierarchical File Naming Systems 29
 - 3.2.3 File and I/O System Calls 31
 - 3.2.4 Open Files 34
 - 3.2.5 Examples of File I/O 36
- 3.3 Information and Meta-Information 39
- 3.4 Naming Operating System Objects 40
- 3.5 Devices as Files 41
 - 3.5.1 Unification of the File and Device Concepts 42
- 3.6 The Process Concept 42
 - 3.6.1 Processes and Programs 43
 - 3.6.2 Process Management System Calls 44
 - 3.6.3 Process Hierarchy 48
- 3.7 Communication between Processes 49
 - 3.7.1 Communication-Related System Calls 50
 - 3.7.2 Example of Interprocess Communication 51
- 3.8 UNIX-Style Process Creation 55
- 3.9 Standard Input and Standard Output 57
 - 3.10 Communicating with Pipes 59
 - 3.10.1 Naming of Pipes and Message Queues 62
- 3.11 Summary of System Call Interfaces 63
- 3.12 Operating System Examples 64
 - 3.12.1 UNIX 64
 - 3.12.2 Mach 65
 - 3.12.3 MS/DOS 65
 - 3.12.4 Windows NT 66
 - 3.12.5 OS/2 66
 - 3.12.6 Macintosh OS 66

- 3.13 The User Interface to an Operating System* 67
 - 3.13.1 Why You Need a Shell 67
 - 3.13.2 The Specification of the Shell 67
 - 3.13.3 Implementing the Shell 67
- 3.14 Summary 71
 - 3.14.1 Terminology 72
 - 3.14.2 Review Questions 73
 - 3.14.3 Further Reading 74
- 3.15 Problems 74

4

Design Techniques I 78

- 4.1 Operating Systems and Design 79
 - 4.1.1 The Design Process 79
 - 4.1.2 Relationship to Software Engineering 80
 - 4.1.3 A Design Example 81
 - 4.1.4 Learning Design through Operating Systems 82
- 4.2 Design Problems 82
 - 4.2.1 Design Skills 83
 - 4.2.2 Design Space 84
 - 4.2.3 Design Levels 85
- 4.3 Design Techniques 86
- 4.4 Two-Level Implementation 86
 - 4.4.1 Overview 86
 - 4.4.2 Motivation 86
 - 4.4.3 Operating System Examples 87
 - 4.4.4 Computer Science Examples 88
 - 4.4.5 Applicability 88
 - 4.4.6 Consequences 89
 - 4.4.7 Implementation Issues and Variations 89
 - 4.4.8 Related Design Techniques 95
- 4.5 Interface Design 95
 - 4.5.1 Overview 95
 - 4.5.2 Motivation 96
 - 4.5.3 Applicability 99
 - 4.5.4 Consequences 99
 - 4.5.5 Related Design Techniques 99
- 4.6 Connection in Protocols 99
 - 4.6.1 Overview 99
 - 4.6.2 Motivation 100
 - 4.6.3 Operating System Examples 101
 - 4.6.4 Computer Science Examples 101
 - 4.6.5 Applicability 102
 - 4.6.6 Consequences 102

- 4.6.7 Implementation Issues and Variations 102
- 4.6.8 Related Design Techniques 104

4.7 Interactive and Programming Interfaces 105

- 4.7.1 Overview 105
- 4.7.2 Motivation 105
- 4.7.3 Operating System Examples 106
- 4.7.4 Computer Science Examples 107
- 4.7.5 Applicability 107
- 4.7.6 Consequences 107
- 4.7.7 Implementation Issues and Variations 107
- 4.7.8 Related Design Techniques 108

4.8 Decomposition Patterns 108

- 4.8.1 Overview 108
- 4.8.2 Motivation 108
- 4.8.3 Operating System Examples 110
- 4.8.4 Computer Science Examples 110
- 4.8.5 Applicability 111
- 4.8.6 Consequences 111
- 4.8.7 Implementation Issues and Variations 111
- 4.8.8 Related Design Techniques 112

4.9 Summary 113

- 4.9.1 Terminology 113
- 4.9.2 Review Questions 114

4.10 Problems 115

5

Implementing Processes 116

- 5.1 The System Call Interface 117
- 5.2 Implementation of a Simple Operating System 119
 - 5.2.1 Guide to the Code 119
 - 5.2.2 The Architecture 121
 - 5.2.3 System Constants 122
 - 5.2.4 Global Data 123
- 5.3 Implementation of Processes 126
 - 5.3.1 Process Creation 126
 - 5.3.2 Process States 128
 - 5.3.3 Process Dispatching 129
 - 5.3.4 The System Stack 132
 - 5.3.5 Timer Interrupts 132
- 5.4 System Initialization 133
 - 5.4.1 The Initial Process 134
- 5.5 Process Switching 136
 - 5.5.1 Switching between Processes 136

5.5.2	Flow of Control	137
5.6	System Call Interrupt Handling	140
5.6.1	Copying Messages between Address Spaces	145
5.7	Program Error Interrupts	146
5.8	Disk Driver Subsystem	146
5.8.1	Communicating with the Disk Controller	149
5.9	Implementation of Waiting	150
5.9.1	Waiting for Messages	150
5.9.2	Waiting inside a System Call	151
5.9.3	Suspending System Calls	152
5.10	Flow of Control through the Operating System	153
5.11	Signaling in an Operating System	156
5.12	Interrupts in the Operating System	157
5.13	Operating Systems as Event and Table Managers	158
5.14	Process Implementation	160
5.14.1	The Process Table and Process Descriptors	160
5.15	Examples of Process Implementation	161
5.16	Monoprogramming*	161
5.16.1	Batch Systems	162
5.16.2	Multiprogramming and I/O Overlap	163
5.16.3	Personal Computer Systems	164
5.17	Summary	167
5.17.1	Terminology	167
5.17.2	Review Questions	168
5.17.3	Further Reading	169
5.18	Problems	169

6

Parallel Systems 175

6.1	Parallel Hardware	176
6.2	An Operating System for a Two-Processor System	177
6.2.1	Using Two Separate Operating Systems	177
6.2.2	Sharing the Operating System	178
6.3	Race Conditions with a Shared Process Table	180
6.4	Atomic Actions	181
6.4.1	Hardware Implementation of Atomic Actions	183
6.5	A Multiprocessor Operating System	183

6.5.1	The Current Process Variable	185
6.5.2	Dispatching With a Shared Process Table	185
6.5.3	Busy Waiting	187
6.5.4	Handling the Queues	187
6.5.5	Grouping of Shared Variables	188
6.5.6	A General Solution	189
6.5.7	Using Two Process Tables	192
6.6	Examples of Multiprocessor Operating Systems	193
6.7	Threads	193
6.7.1	The Thread Concept	193
6.7.2	Thread System Calls	194
6.7.3	Advantages of Threads	195
6.7.4	Uses of Threads	195
6.7.5	Thread Implementation*	197
6.7.6	Splitting the Process Concept	203
6.7.7	Lightweight Processes and User Threads	203
6.7.8	Examples of Threads	204
6.8	Kernel-mode Processes*	205
6.8.1	Data Structures for Kernel-Mode Processes	206
6.8.2	Process Creation with Kernel-Mode Processes	207
6.8.3	Interrupt Handlers for Kernel-Mode Processes	208
6.8.4	Switching Processes for Kernel-Mode Processes	210
6.8.5	How the System Stack is Used	211
6.8.6	Waiting with Kernel-Mode Processes	212
6.8.7	Dispatching with Kernel-Mode Processes	213
6.8.8	Kernel-Mode only Processes	214
6.8.9	Trade-Offs of Kernel-Mode Processes	215
6.8.10	Examples of Kernel-Mode Processes	215
6.9	Implementation of Mutual Exclusion	216
6.9.1	First Solution: Disabling Interrupts	216
6.9.2	Second Solution: Using ExchangeWord	217
6.9.3	Third Solution: Software Solutions	217
6.9.4	When to Use Each Solution	219
6.9.5	Examples of Implementing Mutual Exclusion	219

- 6.10 Varieties of Computer Models* 220
 - 6.10.1 Multiprogramming 221
 - 6.10.2 Multiprocessing 221
- 6.11 Summary 223
 - 6.11.1 Terminology 223
 - 6.11.2 Review Questions 224
 - 6.11.3 Further Reading 225
- 6.13 Problems 225

7

Interprocess Communication Patterns 230

- 7.1 Using Interprocess Communication 231
- 7.2 Patterns of Interprocess Communication 231
 - 7.2.1 Competing and Cooperating 232
- 7.3 Problems When Processes Compete 233
- 7.4 Race Conditions and Atomic Actions 235
- 7.5 New Message-Passing System Calls 238
- 7.6 IPC Pattern: Mutual Exclusion 239
 - 7.6.1 N Process Mutual Exclusion 241
 - 7.6.2 Voluntary Cooperation in Mutual Exclusion 241
- 7.7 IPC Pattern: Signaling 242
- 7.8 IPC Pattern: Rendezvous 243
 - 7.8.1 Many Process Rendezvous 244
- 7.9 IPC Pattern: Producer-Consumer 245
 - 7.9.1 The Basic Producer-Consumer Pattern 247
 - 7.9.2 Limiting the Number of Buffers Used 248
 - 7.9.3 Multiple Producers and Consumers 249
- 7.10 IPC Pattern: Client-Server 250
- 7.11 IPC Pattern: Multiple Servers and Clients* 254
- 7.12 IPC Pattern: Database Access and Update 256
 - 7.12.1 Scheduling 260
 - 7.12.2 Priority 262
 - 7.12.3 Scheduling Queues 262
- 7.13 Review of Interprocess Communication Patterns 262
 - 7.13.1 Mutual Exclusion 262
 - 7.13.2 Signaling 263
 - 7.13.3 Rendezvous 263
 - 7.13.4 Producer-Consumer 263

- 7.13.5 Client-Server 264
- 7.13.6 Multiple Servers and Clients 264
- 7.13.7 Database Access and Update 265
- 7.14 A Physical Analogy 265
- 7.15 Failure of Processes 267
 - 7.15.1 Recovery from Failure 270
- 7.16 Summary 270
 - 7.16.1 Terminology 272
 - 7.16.2 Review Questions 272
 - 7.16.3 Further Reading 273
- 7.17 Problems 273

8

Processes 277

- 8.1 Everyday Scheduling 278
 - 8.1.1 First-Come, First-Served Scheduling 278
 - 8.1.2 Shortest-Job-First Scheduling 278
 - 8.1.3 Highest-Response-Ratio-Next Scheduling 280
 - 8.1.4 Priority Scheduling 281
 - 8.1.5 Deadline Scheduling 281
 - 8.1.6 Round-Robin Scheduling 281
 - 8.1.7 Summary 282
- 8.2 Preemptive Scheduling Methods 282
 - 8.2.1 Scheduling Overview 283
 - 8.2.2 Round-Robin Scheduling 283
 - 8.2.3 Heavily Loaded Systems 285
 - 8.2.4 Two Queues 285
 - 8.2.5 Multiple Queues 286
- 8.3 Policy versus Mechanism in Scheduling 286
- 8.4 A Scheduling Example 288
- 8.5 Scheduling in Real Operating Systems 288
 - 8.5.1 Scheduling in UNIX SVR4 289
 - 8.5.2 Scheduling in Solaris 290
 - 8.5.3 Scheduling in OS/2 2.0 290
 - 8.5.4 Scheduling in Windows NT 3.51 290
 - 8.5.5 Scheduling in Other Operating Systems 290
- 8.6 Deadlock 291
- 8.7 Why Deadlock Is a Problem 293
- 8.8 Conditions for Deadlock to Occur 293
- 8.9 How to Deal with Deadlock 294
 - 8.9.1 Deadlock Prevention 294
 - 8.9.2 Deadlock Avoidance 295
 - 8.9.3 Deadlock Recovery 295

8.10	A Sequence of Approaches to the Deadlock Problem	296
8.11	Two-Phase Locking	296
8.12	Starvation	297
8.13	Message Passing Variations	298
8.13.1	Using PIDs as Message Addresses	298
8.13.2	Message Passing with Nonblocking Receives	298
8.13.3	Message Passing with Blocking Sends	301
8.13.4	Remote Procedure Calls	302
8.14	Synchronization	305
8.14.1	Definition of Synchronization	305
8.14.2	Review of Synchronization	306
8.15	Separating Data Transfer and Synchronization	308
8.16	Semaphores	308
8.16.1	Specification of Semaphore Operations	309
8.16.2	Implementation of Semaphores	310
8.16.3	An Analogy	311
8.16.4	Mutual Exclusion with Semaphores	311
8.16.5	Rendezvous with Semaphores	312
8.16.6	Producer-Consumer (one buffer) with Semaphores	312
8.16.7	Counting Semaphores	313
8.16.8	Producer-Consumer (N buffers) with Semaphores	314
8.16.9	Semaphores and Messages	316
8.17	Implementing Semaphores*	316
8.17.1	System Constants	316
8.18	Using Semaphores in the Simple Operating System	320
8.19	Programming-Language-Based Synchronization Primitives	322
8.19.1	Monitors	323
8.19.2	Synchronization Primitives in Ada 95	328
8.20	Message Passing Design Issues	335
8.20.1	Copying Messages	335
8.20.2	Longer Messages	337
8.21	IPC in Mach	337
8.21.1	Tasks and Threads	337
8.21.2	Ports and Messages	337
8.21.3	Objects	338
8.22	IPC and Synchronization Examples	338

8.22.1	Signals	338
8.22.2	SVR4 UNIX	339
8.22.3	Windows NT	339
8.22.4	OS/2	339
8.22.5	Solaris	340
8.23	Summary	340
8.23.1	Terminology	341
8.23.2	Review Questions	342
8.23.3	Further Reading	343
8.24	Problems	343

9

Design Techniques II 350

9.1	Indirection	350
9.1.1	Overview	350
9.1.2	Motivation	350
9.1.3	Operating System Examples	351
9.1.4	Computer Science Examples	352
9.1.5	Discussion	354
9.1.6	Applicability	355
9.1.7	Consequences	356
9.2	Using State Machines	356
9.2.1	Overview	356
9.2.2	Operating System Example	356
9.2.3	Computer Science Example	356
9.2.4	Applicability	356
9.2.5	Consequences	357
9.2.6	Implementation Issues and Variations	357
9.3	Win Big, Then Give Some Back	358
9.3.1	Overview	358
9.3.2	Motivation	358
9.3.3	Operating System Examples	359
9.3.4	Computer Science Examples	359
9.3.5	Applicability	359
9.3.6	Consequences	359
9.4	Separation of Concepts	360
9.4.1	Overview	360
9.4.2	Motivation	360
9.4.3	Operating System Examples	360
9.4.4	Computer Science Examples	361
9.4.5	Applicability	362
9.4.6	Consequences	362
9.4.7	Implementation Issues and Variations	362
9.4.8	Related Design Techniques	363
9.5	Reducing a Problem to a Special Case	363
9.5.1	Overview	363
9.5.2	Motivation	363

9.5.3	Operating System Examples	363
9.5.4	Computer Science Examples	364
9.5.5	Applicability	364
9.5.6	Consequences	365
9.5.7	Implementation Issues and Variations	365
9.6	Reentrant Programs	365
9.6.1	Overview	365
9.6.2	Motivation	365
9.6.3	Operating System Examples	366
9.6.4	Computer Science Examples	367
9.6.5	Applicability	367
9.6.6	Consequences	367
9.6.7	Implementation Issues and Variations	368
9.6.8	Related Design Techniques	368
9.7	Using Models for Inspiration	368
9.7.1	Overview	368
9.7.2	Motivation	368
9.7.3	Operating System Examples	369
9.7.4	Computer Science Examples	369
9.7.5	Applicability	369
9.7.6	Consequences	369
9.8	Adding a New Facility to a System	369
9.8.1	Overview	369
9.8.2	Motivation	370
9.8.3	Operating System Examples	370
9.8.4	Computer Science Examples	371
9.8.5	Applicability	371
9.8.6	Consequences	372
9.8.7	Related Design Techniques	372
9.9	Summary	373
9.9.1	Terminology	373
9.9.2	Review Questions	374
9.10	Problems	374

10

Memory Management 377

10.1	Levels of Memory Management	378
10.2	Linking and Loading a Process	378
10.2.1	Creating a Load Module	379
10.2.2	Loading a Load Module	387
10.2.3	Allocating Memory in a Running Process	388
10.3	Variations in Program Loading	389
10.3.1	Load Time Dynamic Linking	389
10.3.2	Run Time Dynamic Linking	390

10.4	Why Use Dynamic Memory Allocation?	393
10.5	The Memory Management Design Problem*	394
10.6	Solutions to the Memory Management Design Problem*	395
10.6.1	Static Division into a Fixed Number of Blocks	395
10.6.2	Buddy Systems	397
10.6.3	Powers-of-two Allocation	398
10.7	Dynamic Memory Allocation*	399
10.8	Keeping Track of the Blocks*	400
10.8.1	The List Method	400
10.8.2	Keeping Allocated Blocks on the Block List	401
10.8.3	Where Is the Block List Kept?	402
10.8.4	Using Block Headers as Free List Nodes	403
10.8.5	The Bitmap Method	404
10.8.6	Comparing Free List Methods	405
10.9	Which Free Block to Allocate?*	406
10.10	Examples of Dynamic Memory Allocation	407
10.11	Logical and Physical Memory	408
10.12	Allocating Memory to Processes	409
10.12.1	Static Memory Management	410
10.12.2	Handling Variable-Sized Processes	411
10.13	Multiprogramming Issues	412
10.14	Memory Protection	413
10.15	Memory Management System Calls	414
10.15.1	Static Allocation of Memory to Processes	414
10.15.2	Dynamic Allocation of Memory to Processes	414
10.15.3	What about New and Malloc?	417
10.15.4	Freeing Memory at Each Level	417
10.15.5	A Different Memory Management System Call	419
10.16	Example Code for Memory Allocation*	419
10.17	Summary	423
10.17.1	Terminology	424
10.17.2	Review Questions	425

- 10.17.3 Further Reading 425
- 10.18 Problems 426

11

Virtual Memory 429

- 11.1 Fragmentation and Compaction* 430
- 11.2 Dealing with Fragmentation 430
 - 11.2.1 Separate Code and Data Spaces 431
 - 11.2.2 Segments 431
 - 11.2.3 Noncontiguous Logical Address Spaces 434
 - 11.2.4 Page Tables in Hardware Registers 436
 - 11.2.5 Page Tables in Memory 437
 - 11.2.6 Using a Page Table Cache 438
 - 11.2.7 Analysis Models of Paging with Caching 441
 - 11.2.8 Memory Allocation with Paging 442
 - 11.2.9 Terminology: Page and Page Frame 443
 - 11.2.10 Page Tables 443
 - 11.2.11 Paging Summary 445
- 11.3 Memory Allocation Code with Pages* 446
- 11.4 Sharing the Processor and Sharing Memory* 449
- 11.5 Swapping* 450
 - 11.5.1 Efficient Resource Use and User Needs 451
- 11.6 Overlays* 454
 - 11.6.1 Overlays in PCs 455
- 11.7 Implementing Virtual Memory 457
 - 11.7.1 Hardware Required to Support Virtual Memory 458
 - 11.7.2 Software Required to Support Virtual Memory 459
- 11.8 What is the Cost of Virtual Memory? 461
 - 11.8.1 Paging More Than One Process 462
 - 11.8.2 Locality 462
- 11.9 Virtual Memory Management 464
- 11.10 Daemons and Events 467
- 11.11 File Mapping 469
 - 11.11.1 The System Call Interface 470
 - 11.11.2 An Example of Using File Mapping 471
 - 11.11.3 Advantages of File Mapping 471
 - 11.11.4 Memory and File Mapping on the IBM 801 472

- 11.11.5 File Mapping Examples 474
- 11.12 Summary 474
 - 11.12.1 Terminology 475
 - 11.12.2 Review Questions 476
 - 11.12.3 Further Reading 477
- 11.13 Problems 477

12

Virtual Memory Systems 483

- 12.1 Page Replacement 484
- 12.2 Global Page Replacement Algorithms 484
 - 12.2.1 Measuring the Performance of a Page Replacement Algorithm 484
 - 12.2.2 Optimal Page Replacement 484
 - 12.2.3 Theories of Program Paging Behavior 485
 - 12.2.4 Random Page Replacement 485
 - 12.2.5 First-In, First-Out FIFO Page Replacement 486
 - 12.2.6 Least Recently Used Page Replacement 487
 - 12.2.7 Approximations of LRU 489
 - 12.2.8 Clock Algorithms 490
- 12.3 Page Replacement Examples 493
- 12.4 Local Page Replacement Algorithms 494
 - 12.4.1 What Is a Working Set? 495
 - 12.4.2 Program Phases 495
 - 12.4.3 Variable Resident Set Sizes 497
 - 12.4.4 The Working Set Paging Algorithm 498
 - 12.4.5 Approximating the Working Set 498
 - 12.4.6 WSClock Paging Algorithm 499
- 12.5 Evaluating Paging Algorithms* 501
 - 12.5.1 Methodology for Paging Simulation 501
 - 12.5.2 Some Page Simulation Results 503
- 12.6 Thrashing and Load Control 504
 - 12.6.1 How Thrashing Occurs 504
 - 12.6.2 Load control 505
 - 12.6.3 Swapping 505
 - 12.6.4 Scheduling and Swapping 506
 - 12.6.5 Load Control and Paging Algorithms 507
 - 12.6.6 Predictive Load Control 507
 - 12.6.7 Preloading of Pages 508
- 12.7 Dealing with Large Page Tables 509
 - 12.7.1 What Is the Problem? 509

12.7.2	Two-Level Paging	509
12.7.3	Benefits of Two-Level Paging	511
12.7.4	Problems with Two-Level Paging	511
12.7.5	Software Page Table Lookups	512
12.8	Recursive Address Spaces*	516
12.9	Paging the Operating System Address Space	518
12.9.1	Locking Pages in Memory	519
12.10	Page Size*	519
12.10.1	Reasons for a Large Page Size	519
12.10.2	Reasons for a Small Page Size	520
12.10.3	Clustering Pages	521
12.11	Segmentation	521
12.11.1	What Is a Segment?	521
12.11.2	Virtual Memory with Segmentation	522
12.11.3	Segmentation with Paging	523
12.11.4	History of Segmentation	523
12.11.5	Segment Terminology	524
12.12	Sharing Memory	526
12.12.1	Reasons for Sharing Memory	527
12.12.2	Shared Memory System Calls	527
12.13	Examples of Virtual Memory Systems	528
12.13.1	Swap Area	528
12.13.2	Page Initialization	529
12.13.3	Page Sharing	529
12.13.4	Double-Handed Clock Algorithm	530
12.13.5	Standby Page Lists	531
12.13.6	Clustering Pages	533
12.13.7	File Mapping	533
12.13.8	Portable Virtual Memory Systems	533
12.13.9	Sparse Address Space	534
12.13.10	OS/2 Version 2.0	534
12.13.11	Windows NT	535
12.13.12	Mach and OSF/1	536
12.13.13	System V Release 4	537
12.13.14	Other Systems	537
12.14	Very Large Address Spaces	538
12.15	Summary	538
12.15.1	Terminology	539
12.15.2	Review Questions	540
12.15.3	Further Reading	541
12.16	Problems	541

13

Design Techniques III 547

13.1	Multiplexing	547
13.1.1	Overview	547
13.1.2	Motivation	547
13.1.3	Operating System Examples	549
13.1.4	Computer Science Examples	549
13.1.5	Applicability	550
13.1.6	Consequences	550
13.2	Late Binding	550
13.2.1	Overview	550
13.2.2	Motivation	550
13.2.3	Operating System Examples	551
13.2.4	Computer Science Examples	551
13.2.5	Applicability	552
13.2.6	Consequences	552
13.2.7	Implementation Issues and Variations	553
13.2.8	Related Design Techniques	554
13.3	Static Versus Dynamic	554
13.3.1	Overview	554
13.3.2	Motivation	554
13.3.3	Operating System Examples	554
13.3.4	Computer Science Examples	556
13.3.5	Applicability	557
13.3.6	Consequences	557
13.3.7	Implementation Issues and Variations	558
13.3.8	Related Design Techniques	561
13.4	Space-Time Tradeoffs	561
13.4.1	Overview	561
13.4.2	Motivation	562
13.4.3	Computer Science Examples	564
13.4.4	Applicability	565
13.4.5	Consequences	565
13.4.6	Implementation Issues and Variations	566
13.4.7	Related Design Techniques	566
13.5	Simple Analytic Models	567
13.5.1	Overview	567
13.5.2	Motivation	567
13.5.3	Operating System Examples	568
13.5.4	Applicability	568
13.5.5	Consequences	568
13.5.6	Implementation Issues and Variations	569

13.6	Summary	570
13.6.1	Terminology	570
13.6.2	Review Questions	571
13.7	Problems	571
14		
I/O Devices	573	
14.1	Devices and Controllers	574
14.1.1	Device Controllers	574
14.2	Terminal Devices*	575
14.2.1	Basic Terminals	575
14.2.2	Display Commands	579
14.2.3	Example Display Commands	580
14.2.4	Keyboard Events	582
14.2.5	Terminal Capability Databases	583
14.2.6	Virtual Terminals	585
14.2.7	Terminal Interfaces	587
14.2.8	Mouse Devices	588
14.2.9	Event Streams	588
14.2.10	Varieties of Two-Stage Processing	589
14.2.11	Graphics Terminals	591
14.2.12	Color and Color Maps	592
14.2.13	Command Oriented Graphics	592
14.2.14	X Terminals	592
14.2.15	Terminal Emulators	593
14.2.16	Virtual Terminals and Terminal Emulators	596
14.2.17	PPP: a Network Emulator	596
14.2.18	Modems	596
14.3	Communication Devices*	598
14.3.1	Serial Ports	598
14.3.2	Parallel Ports	599
14.3.3	Ethernet Devices	599
14.3.4	Other Network Devices	601
14.4	Disk Devices	601
14.4.1	Timing of a Disk Access	603
14.4.2	Floppy Disks	604
14.4.3	RAID Devices	604
14.5	Disk Controllers	606
14.6	SCSI Interfaces*	607
14.7	Tape Devices*	608
14.8	CD Devices*	609
14.9	Summary	610
14.9.1	Terminology	610
14.9.2	Review Questions	611

14.9.3	Further Reading	612
14.10	Problems	613

15**I/O Systems 616**

15.1	I/O System Software	617
15.1.1	Device Drivers	617
15.1.2	Device Driver Interfaces	618
15.1.3	The Two Categories of Device Drivers	619
15.1.4	The Block Device Interface	619
15.1.5	The Character Device Interface	620
15.2	Disk Device Driver Access Strategies	621
15.2.1	Handling Disk Requests Efficiently	621
15.2.2	Double Buffering — An Aside	622
15.2.3	A Disk Scheduling Example	622
15.2.4	Sector Scheduling within Cylinder Scheduling	629
15.2.5	Combined Sector and Cylinder Scheduling	630
15.2.6	Real-Life Disk Head Scheduling	631
15.3	Modeling of Disks*	631
15.3.1	A Disk Scheduling Anomaly	631
15.3.2	Cylinder Correlations	633
15.3.3	A More Accurate Disk Model	633
15.4	Device numbers	634
15.5	Unification of Files and I/O Devices	634
15.6	Generalized Disk Device Drivers	636
15.6.1	Partitioning Large Disks	636
15.6.2	Combining Disks into a Large Logical Disk	636
15.6.3	RAM Disks	637
15.6.4	Memory as a Device	638
15.6.5	Pseudo-ttys	640
15.7	Disk Caching	641
15.8	Two-Level Structure of Device Drivers	643
15.9	SCSI Device Drivers	643
15.10	Examples of I/O Systems	644
15.11	Summary	645
15.11.1	Terminology	645
15.11.2	Review Questions	646
15.11.3	Further Reading	647
15.12	Problems	647

16**File Systems 653**

- 16.1 The Need for Files 654
 - 16.1.1 Using Disks Directly for Persistent Storage 654
 - 16.1.2 Files 654
 - 16.1.3 Levels in the File System 654
- 16.2 The File Abstraction 656
 - 16.2.1 Variations on the File Abstraction 656
 - 16.2.2 Logical File Structure 657
 - 16.2.3 File Size and Granularity 658
 - 16.2.4 File Meta-Data 659
- 16.3 File Naming 660
 - 16.3.1 Component Names 660
 - 16.3.2 Directories 660
 - 16.3.3 Path Names 660
 - 16.3.4 Variations and Generalizations 662
 - 16.3.5 File Name Extensions 663
 - 16.3.6 Aliases 664
- 16.4 File System Objects and Operations 665
- 16.5 File System Implementation 668
 - 16.5.1 File System Data Structures 668
 - 16.5.2 File System Organization and Control Flow 670
 - 16.5.3 Connecting to Device Drivers 673
 - 16.5.4 Reading and Writing from Special Files 674
 - 16.5.5 Other File System Calls 674
 - 16.5.6 Avoiding Copying Data 676
 - 16.5.7 Directory Implementation 677
- 16.6 An Example File System Implementation* 679
 - 16.6.1 System Constants and Global Data 679
 - 16.6.2 Disk Cache 680
 - 16.6.3 File Descriptors 682
 - 16.6.4 Open Files 684
 - 16.6.5 Directories 685
 - 16.6.6 File System Initialization 686
 - 16.6.7 File-Related System Calls 686
 - 16.6.8 System Call Procedures 688
- 16.7 Summary 692
 - 16.7.1 Terminology 693
 - 16.7.2 Review Questions 694
 - 16.7.3 Further Reading 694
- 16.8 Problems 694

17**File System Organization 697**

- 17.1 File System Organization 697
 - 17.1.1 What Is a File System? 697
 - 17.1.2 File System Structure 697
 - 17.1.3 The File System Descriptor 699
 - 17.1.4 Variations in File System Layout 699
 - 17.1.5 File Systems in Disk Partitions 700
 - 17.1.6 Combining File Systems 700
 - 17.1.7 Network Mounting of File Systems 702
- 17.2 File Descriptors 703
 - 17.2.1 Where to Keep File Descriptors 703
- 17.3 How File Blocks Are Located on Disk 703
 - 17.3.1 The Block-Mapping Problem 704
 - 17.3.2 Contiguous Files 705
 - 17.3.3 Interleaved Files 706
 - 17.3.4 Keeping a File in Pieces 707
 - 17.3.5 Where to Keep the Disk Block Pointers 708
 - 17.3.6 Disk Block Pointers in the File Descriptor 708
 - 17.3.7 Disk Block Pointers Contiguously on Disk 708
 - 17.3.8 Disk Block Pointers in the Disk Blocks 709
 - 17.3.9 Index Blocks in a Chain 710
 - 17.3.10 Two Levels of Index Blocks 712
 - 17.3.11 Large and Small Files 713
 - 17.3.12 Hybrid Solutions 713
 - 17.3.13 Analogy with Page Tables 714
 - 17.3.14 Inverted Disk Block Indexes 715
 - 17.3.15 Using Larger Pieces 717
 - 17.3.16 Variable-Sized Pieces 718
 - 17.3.17 Disk Compaction 720
- 17.4 Review of File Storage Methods 721
- 17.5 Implementation of the Logical to Physical Block Mapping* 723
- 17.6 File Sizes 725
- 17.7 Booting the Operating System* 726
- 17.8 File System Optimization* 727
 - 17.8.1 Block Size 728
 - 17.8.2 Compressed Files 729
 - 17.8.3 Log-Structured File Systems 729
- 17.9 File System Reliability 730
 - 17.9.1 Backups 730
 - 17.9.2 Consistency Checking 731