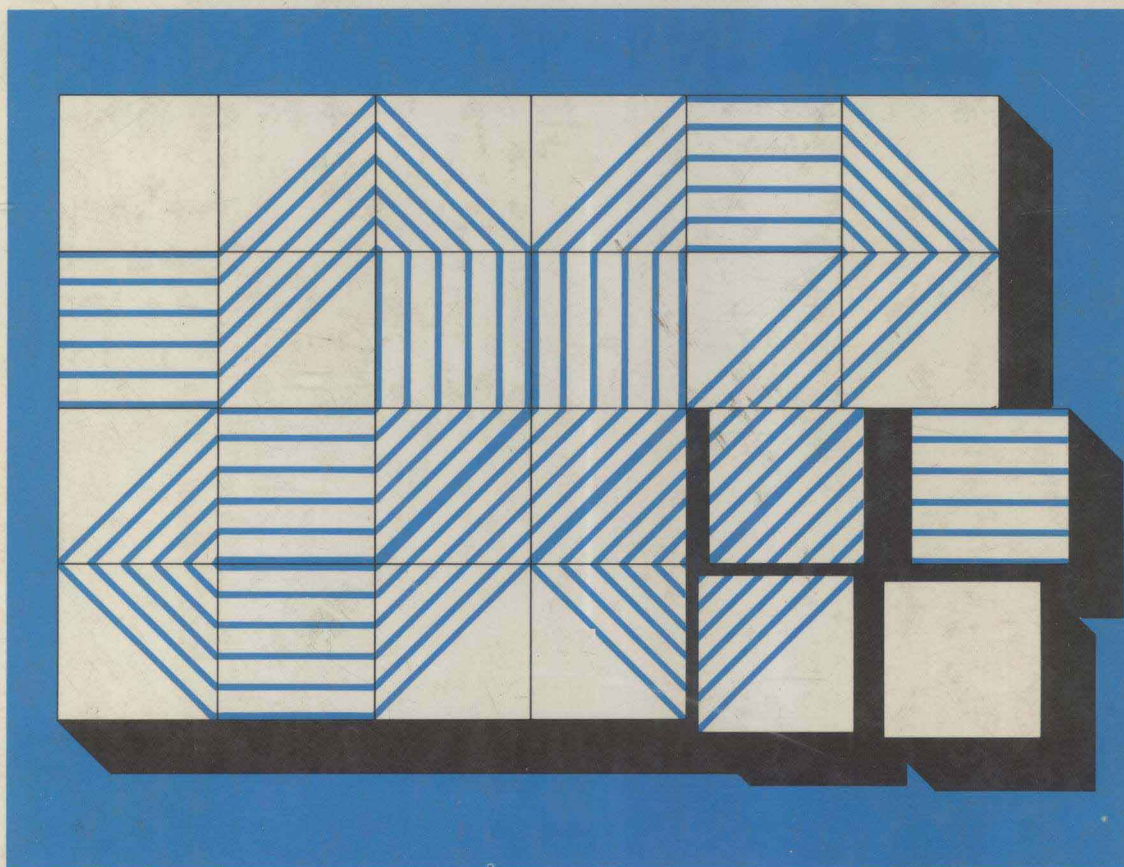


Operating Systems

Advanced Concepts

Maekawa • Oldehoeft • Oldehoeft



Operating Systems

ADVANCED CONCEPTS

Mamoru Maekawa
University of Tokyo

Arthur E. Oldehoeft
Iowa State University

Rodney R. Oldehoeft
Colorado State University



THE BENJAMIN/CUMMINGS PUBLISHING COMPANY, INC.

Menlo Park, California • Reading, Massachusetts
Don Mills, Ontario • Wokingham, U.K. • Amsterdam
Sydney • Singapore • Tokyo • Madrid • Bogota
Santiago • San Juan

Sponsoring Editor: Alan Apt
Production Editors: Laura Kenney, Julie Kranhold
Copyeditor: Carol Dondrea
Illustrator: Lisa Torri
Cover Designer: John Edeen
Compositor: Graphic Typesetting Service

The basic text of this book was designed using the Modular Design System, as developed by Wendy Earl and Design Office Bruce Kortebein.

Copyright © 1987 by The Benjamin/Cummings 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.

The programs presented in this book have been included for their instructional value. They have been tested with care but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs.

Library of Congress Cataloging-in-Publication Data

Maekawa, M. (Mamoru), 1942-
Operating systems.

Includes bibliographies and index.

1. Operating systems (Computers) I. Oldehoeft, Arthur E. II. Oldehoeft, Rodney R. III. Title.
QA76.76.063M335 1987 005.4'3 86-28419
ISBN 0-8053-7121-4

BCDEFGHIJ-DO-8 987

The Benjamin/Cummings Publishing Company, Inc.
2727 Sand Hill Road
Menlo Park, CA 94025

Operating Systems

ADVANCED CONCEPTS

Preface

The study of computer operating systems has progressed from learning an ad hoc collection of folk wisdom about how systems work to examining a coherent set of topics about which substantial theoretical and practical knowledge has been developed. We have been teaching modern operating systems to undergraduates and graduate students for 10 to 15 years at several universities, and believe that introductory material for the undergraduate student is well agreed upon and adequately explicated in several modern texts.

Unfortunately, at the graduate-student or professional level, no such agreement exists. Worse, to teach advanced material properly we have been forced to rely directly on the current literature. Although reference to the contemporary work of researchers and practitioners is useful at this level, students (and instructors as well) have felt the lack of a centralized source of information upon which to rely for an overall framework and source of direction.

The purpose of this book is to provide that needed resource. We have identified eight major subjects in the design and analysis of modern operating systems and have brought together our work on each topic to give the reader a modern course in the advanced topics of importance. The synthesis of many points of view gives the reader an accurate picture of the current state of each topic. The subjects are enduring ones that are not dependent on a particular vendor's system or on current technology; this material should be useful in design and analysis of future systems as well.

Operating Systems: Advanced Concepts continues to rely on original sources for two reasons. First, to keep the book to a manageable size, some subjects are covered completely at a fundamental level with references to generalizations and special cases that the interested reader may pursue. Second, we believe that at this level the serious student of operating systems must be cognizant of the contemporary literature; we hope this approach will help train the reader in professional reading habits if they are not already established. To make this easy, we have often chosen to refer to the most accessible items of literature (major journal or conference articles) instead of the earliest sources (often internal technical reports). In addition to cited references, we provide an additional bibliography of literature for each topic.

Our appreciation is due to Misses Noriko Shiogama and Nahoko Muraio, who entered the first handwritten version of the manuscript. The following reviewers have carefully read and improved the book through several drafts: Imtiaz Ahmad, University of Windsor; Rob Cubert, Sacramento State University; D. M. Etter, University of New Mexico; William Franta, University of Minnesota; Dale Grit, Colorado State University; Teofilo Gonzales, University of Texas at Dallas; Evan Ivie,

v

Brigham Young University; Roger King, University of Colorado; Marek Rusinkiewicz, University of Houston; and Nish Thakor, Northwestern University. Many students in our operating classes have been exposed to this material and have selflessly pointed out areas for clarification and improvement. All of these people have made this an enjoyable project and, we hope, a valuable product. Finally, we would like to acknowledge our families and friends who have provided support and encouragement over the course of the project.

*Mamoru Maekawa
Arthur E. Oldehoeft
Rodney R. Oldehoeft*

Acknowledgments

Figure 3.12 is reprinted from *Ada for Experienced Programmers* by A. Nico Habermann and Dwane E. Perry (Addison-Wesley, 1983).

Figures 6.21 and 6.22 and Tables 6.3 and 6.4 are reprinted from Mamoru Maekawa, “A N Algorithm for Mutual Exclusion in Decentralized Systems,” in *ACM Transactions on Computer Systems* 3 (May 1985). Reprinted by permission of the Association for Computing Machinery, Inc.

Figure 8.24 is reprinted from Voydock and Kent, “Security Mechanisms in High-Level Network Protocols,” in *Computing Surveys* 15 (1983). Reprinted by permission of the Association for Computing Machinery, Inc.

Contents

1	<i>Overview</i>	<i>1</i>
1.1	Introduction	1
1.2	Chapter 2: Process Synchronization	1
1.3	Chapter 3: Language Mechanism for Concurrency	2
1.4	Chapter 4: Deadlock	2
1.5	Chapter 5: Virtual Memory	3
1.6	Chapter 6: Distributed Systems	3
1.7	Chapter 7: Distributed Concurrency Control, Deadlock, and Recovery	4
1.8	Chapter 8: Computer Security	4
1.9	Chapter 9: Queuing Models of Computer Systems	5
	References	6
2	<i>Process Synchronization</i>	<i>9</i>
2.1	Background and Review	9
2.1.1	Concept of a Process	9
2.1.2	Concurrent Processes	11
	THE MODEL	12
	DETERMINATE SYSTEMS OF PROCESSES	12
	MUTUALLY NONINTERFERING SYSTEMS	13
	MAXIMALLY PARALLEL SYSTEMS	14
2.1.3	Critical Sections	15
2.1.4	Early Mechanisms	15
2.1.5	Semaphores	16
2.1.6	Common Synchronization Problems	17
2.2	Sequencers and Eventcounts	17
2.2.1	The Producer/Consumer Problem Using Eventcounts	20
2.2.2	The Reader/Writer Problem Using Eventcounts	20

2.2.3	Tagged Messages	21
2.3	OR Synchronization	23
2.4	AND Synchronization	24
2.4.1	Dining Philosopher Problem	30
2.4.2	Starvation and Efficiency	30
2.4.3	Cigarette Smoker's Problem	31
2.5	NOT Synchronization	31
2.6	Cooperation Without Mutual Exclusion	34
2.6.1	Lamport's Theorem	35
2.6.2	Single Writer/Multiple Reader Problem	36
2.7	Interprocess Communication	37
2.7.1	Direct and Indirect Communication	38
2.7.2	Capacity	40
2.7.3	Message Size	40
2.8	Summary of Mechanisms	41
2.9	Issues of Design and Implementation	42
2.9.1	Overhead	42
FIRMWARE IMPLEMENTATIONS		42
PRIVATE SEMAPHORES		43
2.9.2	Real-Time and Fault Tolerance	43
2.9.3	Creation and Deletion of Semaphores and Sequencers	45
2.9.4	Misuse of Mechanisms	46
2.10	Summary	46
KEY WORDS		46
QUESTIONS		47
PROBLEMS		49
REFERENCES		52
SUGGESTED READINGS		53
3	<i>Language Mechanisms for Concurrency</i>	55
3.1	Introduction	55
3.2	The Object Model and Monitors	55
3.2.1	Mechanisms	57
3.2.2	The Reader/Writer Problem	59
3.2.3	A Disk-Head Scheduler	62
3.2.4	Design and Usage Issues	62
PRIORITIES		63
NESTED MONITOR CALLS		64

3.3	Analysis Framework for Mechanisms	64
3.3.1	Requirements	64
	APPLICABILITY TO CENTRALIZED AND DISTRIBUTED SYSTEMS	65
	EXPRESSIVE POWER	65
	MODULARITY	65
	EASE OF USE	66
	PROGRAM STRUCTURE	66
	PROCESS FAILURES AND TIME-OUTS	67
	UNANTICIPATED FAULTS AND RECOVERY	67
	REAL-TIME SYSTEMS	68
3.3.2	Languages and Systems	68
3.4	Some Concurrent Programming Mechanisms	69
3.4.1	Serializers	69
3.4.2	Path Expressions	71
	OPEN PATH EXPRESSIONS AND PATH PASCAL	72
	IMPLEMENTATION	74
	PREDICATE PATH EXPRESSIONS	77
3.4.3	Communicating Sequential Processes	79
3.4.4	Distributed Processes	81
3.5	Ada Concurrent Programming Mechanisms	82
3.5.1	Task Declaration and Initiation	83
3.5.2	Entries and the Accept Statement	84
3.5.3	The Select Statement	85
3.5.4	Real-Time Processing	88
3.5.5	Termination and Exceptions	90
3.5.6	Ada and Concurrent Programming Requirements	91
3.6	Summary	92
	KEY WORDS	93
	QUESTIONS	93
	PROBLEMS	94
	REFERENCES	97
	SUGGESTED READINGS	99
4	<i>Deadlock</i>	101
4.1	Introduction	101
4.2	The Deadlock Problem	101
4.2.1	Definition of Deadlock	101
4.2.2	Examples of Deadlock	104
4.2.3	Resource Types	105
4.2.4	Deadlock Policies	105

4.3	Concepts from Graph Theory	106
4.4	The General Model	108
4.4.1	General Resource Graph	108
4.4.2	Operations on Resources	109
4.4.3	Necessary and Sufficient Conditions for Deadlock	110
4.5	Special Cases with Useful Results	115
4.5.1	Single-Unit Requests	115
4.5.2	Consumable Resources Only	117
4.5.3	Reusable Resources Only	118
	DETECTION IN REUSEABLE RESOURCE SYSTEMS	118
	AVOIDANCE IN REUSABLE RESOURCE SYSTEMS	121
4.6	Recovery from Deadlock	122
4.7	Prevention by System Design	124
4.8	Total System Design	125
4.9	Summary	126
	KEY WORDS	126
	QUESTIONS	127
	PROBLEMS	129
	REFERENCES	130
	SUGGESTED READINGS	130

5 *Virtual Memory* 133

5.1	Introduction	133
5.2	Background and Review	133
5.2.1	Hardware Support	134
5.2.2	Page-Fault Rate and Principle of Locality	134
5.2.3	Software Components	136
	OPERATING SYSTEM POLICIES	136
	PROCESS BEHAVIOR	139
5.3	Stack Algorithms	140
5.3.1	Cost Function	140
5.3.2	Definition of a Stack Algorithm	141
5.3.3	Stack-Updating Procedure	142
5.3.4	Calculating Cost Function	144
5.3.5	The Extension Problem	146
5.4	Working Sets	147
5.4.1	Definition of Working Set	148
5.4.2	Properties of Working Sets	149

5.4.3	Implementation	150
5.5	Models of Virtual Memory	151
5.5.1	An Extrinsic Model—Lifetime Curves	151
5.5.2	An Intrinsic Model—LRU Stack	153
5.6	Clock Algorithms	153
5.6.1	A Working Set Approximation—WSClock	154
5.6.2	Load-Control Methods	154
	LT/RT LOAD CONTROL	154
	WSCLOCK LOAD CONTROL	156
	CLOCK LOAD CONTROL	156
	CHOOSING A PROCESS TO DEACTIVATE	157
5.6.3	Simulation Results	157
	PROCESS DEACTIVATION POLICIES	157
	LT/RT CONTROL	158
	CLOCK LOAD CONTROL	158
	RELATIVE PERFORMANCE	158
5.7	Program Restructuring	158
5.7.1	Theoretical Bounds on Paging Performance	159
	PAGING PERFORMANCE MODEL	159
	SECTORING PERFORMANCE MODEL	160
	A LOWER BOUND	160
	AN UPPER BOUND	161
5.7.2	An Experiment	161
	BUILDING THE INTERSECTOR REFERENCE MATRIX	163
	CLUSTERING PROCEDURE	163
	EXPERIMENTAL RESULTS	164
5.8	Summary	165
	KEY WORDS	165
	QUESTIONS	166
	PROBLEMS	168
	REFERENCES	171
	SUGGESTED READINGS	172

6 *Distributed Systems* 177

6.1	Introduction	177
6.2	Layered Structures	178
6.2.1	The Reference Model of Open Systems Interconnection	178
6.2.2	The Local Area Network Reference Model	183
6.6.3	Implementation Strategies	183

6.3	The First Three Layers	184
6.3.1	The Physical Layer	184
	TELEPHONE SYSTEMS	185
	COMMUNICATION SATELLITES	187
	LOCAL AREA NETWORKS	187
6.3.2	The Data-Link Layer	188
6.3.3	The Network Layer	190
6.3.4	Multicomputer Organizations	192
6.4	The Middle Two Layers	194
6.4.1	The Transport Layer	194
6.4.2	The Session Layer	200
6.5	Proprietary Network Architectures	200
6.6	Distributed Process Management	201
6.6.1	Issues in Distributed Algorithms	202
	THE CHARACTERIZATION OF DISTRIBUTED ALGORITHMS	202
	ORDERING OF EVENTS IN A DISTRIBUTED SYSTEM	204
6.6.2	Requirements for Distributed Mutual Exclusion Algorithms	205
6.7	Lamport's Algorithm	207
6.8	Richart and Agrawala's Algorithm	208
6.9	Maekawa's Square-Root Algorithm	211
6.9.1	Other Improvement Techniques	211
6.9.2	Theoretical and Conceptual Basis for Maekawa's Algorithm	212
6.9.3	Implementation Details of Maekawa's Algorithm	215
6.9.4	An Example Using Maekawa's Algorithm	220
6.9.5	A Comparative Message Traffic Analysis	221
6.9.6	Formation of Member Sets S_i	223
	METHOD 1	224
	METHOD 2	224
6.10	Miscellaneous Considerations	225
6.10.1	Special Network Topologies	225
6.10.2	Management of Message Sequence Numbers	225
6.10.3	Dynamic Changes to the Network Topology	226
6.10.4	Elimination Algorithms	226
6.10.5	Other Approaches	227
6.11	A Comparative Order Analysis	227

6.12 Summary	228
KEY WORDS	229
QUESTIONS	230
PROBLEMS	232
REFERENCES	233
SUGGESTED READINGS	235

7 *Distributed Concurrency Control, Deadlock, and Recovery* 239

7.1 Introduction	239
7.2 Database Consistency	239
7.3 Assumptions and Requirements	240
7.3.1 Database Model	240
7.3.2 Requirements of the Solution	243
7.4 Concurrency Control Based on Locking	244
7.4.1 Lock Actions	244
7.4.2 Structural Properties of Transactions	244
7.4.3 Schedules	246
7.4.4 Serial and Legal Schedules	246
7.4.5 Equivalent Schedules	247
7.5 Theorems on Consistency	249
7.6 Deadlock Detection	251
7.6.1 Centralized Deadlock Detection	253
7.6.2 Hierarchical Deadlock Detection	257
7.6.3 Distributed Deadlock Detection	257
A PERIODIC DEADLOCK-DETECTION ALGORITHM	260
A CONTINUOUS DEADLOCK-DETECTION ALGORITHM	264
7.6.4 Performance	269
7.7 Deadlock Prevention and Avoidance	269
7.8 Lock Granularity	270
7.8.1 Hierarchical Locks	270
7.8.2 Directed Acyclic Graph of Locks	273
7.8.3 Equivalence of the DAG Lock Protocol and Conventional Lock Protocol	274
7.9 Deadlock Freedom Using Edge Locks	276
7.10 Recovery	277
7.10.1 Atomic Actions	278

7.10.2	Implementation of Atomic Actions	278
	LOCK REQUIREMENTS	279
	STORAGE MANAGEMENT FOR ATOMIC ACTIONS	280
	THE TWO-PHASE COMMIT PROTOCOL	281
7.10.3	Structure of Recovery Management System	285
7.11	Synchronization Techniques Based on TimeStamp Ordering	285
7.11.1	Basic Timestamp Ordering Implementation	286
7.11.2	Two-Phase Commit	287
7.11.3	Improvements on the Method of Basic TimeStamp Ordering	289
	THE THOMAS WRITE RULE	290
	MULTIVERSION TIMESTAMP ORDERING	290
	WAIT_DIE AND WOUND_WAIT	290
	CONSERVATIVE TIMESTAMP ORDERING	290
7.12	Summary	292
	KEY WORDS	292
	QUESTIONS	293
	PROBLEMS	294
	REFERENCES	296
	SUGGESTED READINGS	297

8 *Computer Security* 301

8.1	Introduction	301
8.2	Definition of Security and Common Violations	301
8.3	A Model for Access Control	304
8.3.1	Access Matrix Model	304
	REPRESENTATION OF THE PROTECTION STATE	305
	ENFORCEMENT OF ACCESS CONSTRAINTS	306
	PROTECTION STATE TRANSITIONS	306
8.3.2	Levels of Sharing	308
8.3.3	The General Issue of Trust	311
8.3.4	The Confinement Problem	312
8.4	Flow-Secure Access Controls	313
8.5	Information Flow Control	317
8.5.1	The Lattice Model and Security Requirements	317
8.5.2	Types of Information Flow	318

8.5.3	Compile-Time Certification with Static Binding	318
	THE CERTIFICATION PROCESS FOR ELEMENTARY PROGRAMS	319
	HANDLING ARRAY AND RECORD OBJECTS	320
	FLOW ANALYSIS OF PROCEDURE CALLS	321
	EXCEPTION HANDLING	322
	THE CONFINEMENT PROBLEM REVISITED	322
8.5.4	Run-Time Certification with Static Binding	323
8.5.5	Certification with Dynamic Binding	323
8.6	Implementation of Access Controls	323
8.6.1	Implementation Issues	324
	DESIGN PRINCIPLES	324
	ENFORCEMENT OF SPECIFIC SECURITY POLICIES	324
	REPRESENTATION AND MANAGEMENT OF THE ACCESS MATRIX	325
8.6.2	Access Hierarchies	326
8.6.3	Capability Systems	327
	IMPLEMENTATION OF ABSTRACT DATA TYPES	330
	CAPABILITY-BASED ADDRESSING	331
	REVOCAION	333
8.6.4	Access Control List Systems	334
8.6.5	Lock/Key Systems	336
8.6.6	Security Kernels	336
8.6.7	Extensions to Database Management Systems	336
8.6.8	Test and Verification	337
8.7	Cryptography	338
8.7.1	Application of Cryptography to Secure Computer Systems	338
8.7.2	Cryptosystems and Security Requirements	339
8.7.3	Conventional Cryptosystems	341
8.7.4	The Data Encryption Standard (DES)	342
8.7.5	Public-Key Cryptosystems	343
8.7.6	Authentication and Digital Signatures	346
8.7.7	The Use of Encryption for Stored Information	347
8.8	User Authentication	348
8.9	Summary	350
	APPENDIX: Details of the DES Algorithm	
	Initial Permutation and Its Inverse	350
	The Cipher Function f	350
	Key Schedule Calculation	353
	KEY WORDS	356
	QUESTIONS	358
	PROBLEMS	359
	REFERENCES	368
	SUGGESTED READINGS	371

9	<i>Queuing Models of Computer Systems</i>	375
9.1	Introduction	375
9.2	Dynamic Behavior of a Single Queue	376
9.2.1	Performance Measures	377
9.2.2	The Poisson Distribution	377
	POISSON POSTULATES	377
	DEVELOPMENT OF THE POISSON DISTRIBUTION	378
	DISTRIBUTION OF INTERARRIVAL TIMES	379
	AGGREGATION AND BRANCHING OF POISSON STREAMS	379
9.2.3	Analysis of a Single M/M/1 Queue	380
	STEADY-STATE QUEUE-LENGTH PROBABILITIES	380
	PERFORMANCE MEASURES	382
9.2.4	Generalizations on the Single Queue	382
	M/M/c QUEUE	382
	M/M/∞ QUEUE	383
	M/G/1 QUEUE	383
9.3	Open Networks of Queues	385
9.4	Closed Networks—Normalization Constant Method	388
9.4.1	Efficient Computation of the Normalization Constant	389
9.4.2	Queue-Length Probabilities for LD Queues	390
9.4.3	Performance Measures	392
9.4.4	Generalizations	393
9.5	Closed Networks—Mean Value Analysis	394
9.6	Operational Analysis of Queuing Networks	397
9.6.1	Operational Quantities	398
9.6.2	Job Flow Analysis	399
9.6.3	System Response Time	400
9.6.4	Bottleneck Analysis	400
9.6.5	Generalizations	404
9.7	Summary	405
	KEY WORDS	405
	QUESTIONS	406
	PROBLEMS	407
	REFERENCES	408
	SUGGESTED READINGS	409
	Index	411

1.1 Introduction

The purpose of this book is to serve as a resource in the advanced study of modern computer operating systems. The book can be used in at least three ways. First, it can serve as a textbook in a formal course in advanced operating systems for students who have mastered fundamental material in an undergraduate course. We include here advanced-level material on familiar topics (synchronization, deadlock, virtual memory), as well as material that is not generally covered at an elementary level (security, distributed systems and control, modeling and analysis). There is enough material for a two-quarter sequence, and more than enough for a single semester. There are numerous questions and problems. The former will help students review the chapter; the latter should help them delve more deeply into the material of each chapter. Each chapter also includes a list of important terms to help ensure subject mastery.

Second, the book is useful as an organized course for professionals or for self-study. The chapters are organized so that a brief but adequate review precedes material at an advanced level. This will allow the professional whose background is strong but incomplete to quickly “come up to speed” on a particular subject. Where appropriate, references are given to background material that may be valuable for the reader who is working independently.

Third, the book is a guide to current research and methodology for operating system designers. The individual chapters are independent of each other and well suited for the reader who wants to study a particular subject in depth.

The chapters each center on a major topic in the advanced study of operating systems. The following briefly describes the contents of each chapter. The references in this chapter are the basis on which most significant topics in this book are based.

1.2 Chapter 2: Process Synchronization

In this chapter we cover methods for process management that are more advanced than those found in an introductory course. After a brief review of processes and their synchronization via elementary methods, we describe a model that is valuable for analyzing sets of concurrent processes [Bernstein, 1966]. With this model we can demonstrate safe, deterministic execution while ensuring the maximum potential for parallel processing. Advanced methods of process synchronization are surveyed