

**MODERN
OPERATING SYSTEMS**

SECOND EDITION

Other bestselling titles by Andrew S. Tanenbaum

Structured Computer Organization, 4th edition

This widely-read classic, now in its fourth edition, provides the ideal introduction to computer architecture. It covers the topic in an easy-to-understand way, bottom up. There is a chapter on digital logic for beginners, followed by chapters on microarchitecture, the instruction set architecture level, operating systems, assembly language, and parallel computer architectures.

Computer Networks, 3rd edition

This widely-read classic, now in its third edition, provides the ideal introduction to today's and tomorrow's networks. It explains in detail how modern networks are structured. Starting with the physical layer and working up to the application layer, the book covers a vast number of important topics, including wireless communication, fiber optics, data link protocols, Ethernet, routing algorithms, network performance, security, DNS, electronic mail, USENET news, the World Wide Web, and multimedia. The book has especially thorough coverage of TCP/IP and the Internet.

Operating Systems: Design and Implementation, 2nd edition

This popular text on operating systems is the only book covering both the principles of operating systems and their application to a real system. All the traditional operating systems topics are covered in detail. In addition, the principles are carefully illustrated with MINIX, a free POSIX-based UNIX-like operating system for personal computers. Each book contains a free CD-ROM containing the complete MINIX system, including all the source code. The source code is listed in an appendix to the book and explained in detail in the text.

Distributed Operating Systems

This text covers the fundamental concepts of distributed operating systems. Key topics include communication and synchronization, processes and processors, distributed shared memory, distributed file systems, and distributed real-time systems. The principles are illustrated using four chapter-long examples.

MODERN OPERATING SYSTEMS

SECOND EDITION

ANDREW S. TANENBAUM

*Vrije Universiteit
Amsterdam, The Netherlands*



PRENTICE HALL

UPPER SADDLE RIVER, NEW JERSEY 07458

Library of Congress Cataloging in Publication Data

TANENBAUM, ANDREW, S. (date)
Structured Computer Organization

Bibliography

1. Electronic digital computers—Programming

I. Title

QA76.6.T38 001.6'42 74-30322

ISBN 0-13-000000-0

© 2001 by Prentice-Hall, Inc., Upper Saddle River, NJ

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.

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LTD., *Sydney*
EDITORA PRENTICE-HALL DO BRAZIL, LTDA., *Rio de Janeiro*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LTD., *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LTD., *Wellington, New Zealand*

To Suzanne, Barbara, Marvin, and the memory of Bram and Sweetie π

CONTENTS

PREFACE	xvi
1 INTRODUCTION	1
1.1. WHAT IS AN OPERATING SYSTEM? 3	
1.1.1. The Operating System as an Extended Machine 3	
1.1.2. The Operating System as a Resource Manager 5	
1.2. HISTORY OF OPERATING SYSTEMS 6	
1.2.1. The First Generation (1945-55) 6	
1.2.2. The Second Generation (1955-65) 7	
1.2.3. The Third Generation (1965-1980) 9	
1.2.4. The Fourth Generation (1980-Present) 13	
1.2.5. Ontogeny Recapitulates Phylogeny 16	
1.3. THE OPERATING SYSTEM ZOO 18	
1.3.1. Mainframe Operating Systems 18	
1.3.2. Server Operating Systems 19	
1.3.3. Multiprocessor Operating Systems 19	
1.3.4. Personal Computer Operating Systems 19	
1.3.5. Real-Time Operating Systems 19	
1.3.6. Embedded Operating Systems 20	
1.3.7. Smart Card Operating Systems 20	

- 1.4. COMPUTER HARDWARE REVIEW 20
 - 1.4.1. Processors 21
 - 1.4.2. Memory 23
 - 1.4.3. I/O Devices 28
 - 1.4.4. Buses 31

- 1.5. OPERATING SYSTEM CONCEPTS 34
 - 1.5.1. Processes 34
 - 1.5.2. Deadlocks 36
 - 1.5.3. Memory Management 37
 - 1.5.4. Input/Output 38
 - 1.5.5. Files 38
 - 1.5.6. Security 41
 - 1.5.7. The Shell 41
 - 1.5.8. Recycling of Concepts 43

- 1.6. SYSTEM CALLS 44
 - 1.6.1. System Calls for Process Management 48
 - 1.6.2. System Calls for File Management 50
 - 1.6.3. System Calls for Directory Management 51
 - 1.6.4. Miscellaneous System Calls 53
 - 1.6.5. The Windows Win32 API 53

- 1.7. OPERATING SYSTEM STRUCTURE 56
 - 1.7.1. Monolithic Systems 56
 - 1.7.2. Layered Systems 57
 - 1.7.3. Virtual Machines 59
 - 1.7.4. Exokernels 61
 - 1.7.5. Client-Server Model 61

- 1.8. RESEARCH ON OPERATING SYSTEMS 63

- 1.9. OUTLINE OF THE REST OF THIS BOOK 65

- 1.10. METRIC UNITS 66

- 1.11. SUMMARY 67

2 PROCESSES AND THREADS**71**

- 2.1. PROCESSES 71
 - 2.1.1. The Process Model 72
 - 2.1.2. Process Creation 73
 - 2.1.3. Process Termination 75
 - 2.1.4. Process Hierarchies 76
 - 2.1.5. Process States 77
 - 2.1.6. Implementation of Processes 79
- 2.2. THREADS 81
 - 2.2.1. The Thread Model 81
 - 2.2.2. Thread Usage 85
 - 2.2.3. Implementing Threads in User Space 90
 - 2.2.4. Implementing Threads in the Kernel 93
 - 2.2.5. Hybrid Implementations 94
 - 2.2.6. Scheduler Activations 94
 - 2.2.7. Pop-Up Threads 96
 - 2.2.8. Making Single-Threaded Code Multithreaded 97
- 2.3. INTERPROCESS COMMUNICATION 100
 - 2.3.1. Race Conditions 100
 - 2.3.2. Critical Regions 102
 - 2.3.3. Mutual Exclusion with Busy Waiting 103
 - 2.3.4. Sleep and Wakeup 108
 - 2.3.5. Semaphores 110
 - 2.3.6. Mutexes 113
 - 2.3.7. Monitors 115
 - 2.3.8. Message Passing 119
 - 2.3.9. Barriers 123
- 2.4. CLASSICAL IPC PROBLEMS 124
 - 2.4.1. The Dining Philosophers Problem 125
 - 2.4.2. The Readers and Writers Problem 128
 - 2.4.3. The Sleeping Barber Problem 129
- 2.5. SCHEDULING 132
 - 2.5.1. Introduction to Scheduling 132
 - 2.5.2. Scheduling in Batch Systems 138
 - 2.5.3. Scheduling in Interactive Systems 142
 - 2.5.4. Scheduling in Real-Time Systems 148
 - 2.5.5. Policy versus Mechanism 149
 - 2.5.6. Thread Scheduling 150

- 2.6. RESEARCH ON PROCESSES AND THREADS 151
- 2.7. SUMMARY 152

3 DEADLOCKS 159

- 3.1. RESOURCES 160
 - 3.1.1. Preemptable and Nonpreemptable Resources 160
 - 3.1.2. Resource Acquisition 161
- 3.2. INTRODUCTION TO DEADLOCKS 163
 - 3.2.1. Conditions for Deadlock 164
 - 3.2.2. Deadlock Modeling 164
- 3.3. THE OSTRICH ALGORITHM 167
- 3.4. DEADLOCK DETECTION AND RECOVERY 168
 - 3.4.1. Deadlock Detection with One Resource of Each Type 168
 - 3.4.2. Deadlock Detection with Multiple Resource of Each Type 171
 - 3.4.3. Recovery from Deadlock 173
- 3.5. DEADLOCK AVOIDANCE 175
 - 3.5.1. Resource Trajectories 175
 - 3.5.2. Safe and Unsafe States 176
 - 3.5.3. The Banker's Algorithm for a Single Resource 178
 - 3.5.4. The Banker's Algorithm for Multiple Resources 179
- 3.6. DEADLOCK PREVENTION 180
 - 3.6.1. Attacking the Mutual Exclusion Condition 180
 - 3.6.2. Attacking the Hold and Wait Condition 181
 - 3.6.3. Attacking the No Preemption Condition 182
 - 3.6.4. Attacking the Circular Wait Condition 182
- 3.7. OTHER ISSUES 183
 - 3.7.1. Two-Phase Locking 183
 - 3.7.2. Nonresource Deadlocks 184
 - 3.7.3. Starvation 184
- 3.8. RESEARCH ON DEADLOCKS 185
- 3.9. SUMMARY 185

4 MEMORY MANAGEMENT**189**

- 4.1. BASIC MEMORY MANAGEMENT 190
 - 4.1.1. Monoprogramming without Swapping or Paging 190
 - 4.1.2. Multiprogramming with Fixed Partitions 191
 - 4.1.3. Modeling Multiprogramming 192
 - 4.1.4. Analysis of Multiprogramming System Performance 194
 - 4.1.5. Relocation and Protection 194
- 4.2. SWAPPING 196
 - 4.2.1. Memory Management with Bitmaps 199
 - 4.2.2. Memory Management with Linked Lists 200
- 4.3. VIRTUAL MEMORY 202
 - 4.3.1. Paging 202
 - 4.3.2. Page Tables 205
 - 4.3.3. TLBs—Translation Lookaside Buffers 211
 - 4.3.4. Inverted Page Tables 213
- 4.4. PAGE REPLACEMENT ALGORITHMS 214
 - 4.4.1. The Optimal Page Replacement Algorithm 215
 - 4.4.2. The Not Recently Used Page Replacement Algorithm 216
 - 4.4.3. The First-In, First-Out 217
 - 4.4.4. The Second Chance Page Replacement Algorithm 217
 - 4.4.5. The Clock Page Replacement Algorithm 218
 - 4.4.6. The Least Recently Used 218
 - 4.4.7. Simulating LRU in Software 220
 - 4.4.8. The Working Set Page Replacement Algorithm 222
 - 4.4.9. The WSClock Page Replacement Algorithm 225
 - 4.4.:. Summary of Page Replacement Algorithms 227
- 4.5. MODELING PAGE REPLACEMENT ALGORITHMS 228
 - 4.5.1. Belady's Anomaly 229
 - 4.5.2. Stack Algorithms 229
 - 4.5.3. The Distance String 232
 - 4.5.4. Predicting Page Fault Rates 233
- 4.6. DESIGN ISSUES FOR PAGING SYSTEMS 234
 - 4.6.1. Local versus Global Allocation Policies 234
 - 4.6.2. Load Control 236
 - 4.6.3. Page Size 237
 - 4.6.4. Separate Instruction and Data Spaces 239

- 4.6.5. Shared Pages 239
- 4.6.6. Cleaning Policy 241
- 4.6.7. Virtual Memory Interface 241
- 4.7. IMPLEMENTATION ISSUES 242
 - 4.7.1. Operating System Involvement with Paging 242
 - 4.7.2. Page Fault Handling 243
 - 4.7.3. Instruction Backup 244
 - 4.7.4. Locking Pages in Memory 246
 - 4.7.5. Backing Store 246
 - 4.7.6. Separation of Policy and Mechanism 247
- 4.8. SEGMENTATION 249
 - 4.8.1. Implementation of Pure Segmentation 253
 - 4.8.2. Segmentation with Paging: MULTICS 254
 - 4.8.3. Segmentation with Paging: The Intel Pentium 257
- 4.9. RESEARCH ON MEMORY MANAGEMENT 262
- 4.10. SUMMARY 262

5 INPUT/OUTPUT

269

- 5.1. PRINCIPLES OF I/O HARDWARE 269
 - 5.1.1. I/O Devices 270
 - 5.1.2. Device Controllers 271
 - 5.1.3. Memory-Mapped I/O 272
 - 5.1.4. Direct Memory Access 276
 - 5.1.5. Interrupts Revisited 279
- 5.2. PRINCIPLES OF I/O SOFTWARE 282
 - 5.2.1. Goals of the I/O Software 283
 - 5.2.2. Programmed I/O 284
 - 5.2.3. Interrupt-Driven I/O 286
 - 5.2.4. I/O Using DMA 287
- 5.3. I/O SOFTWARE LAYERS 287
 - 5.3.1. Interrupt Handlers 287
 - 5.3.2. Device Drivers 289

- 5.3.3. Device-Independent I/O Software 292
- 5.3.4. User-Space I/O Software 298
- 5.4. DISKS 300
 - 5.4.1. Disk Hardware 300
 - 5.4.2. Disk Formatting 315
 - 5.4.3. Disk Arm Scheduling Algorithms 318
 - 5.4.4. Error Handling 322
 - 5.4.5. Stable Storage 324
- 5.5. CLOCKS 327
 - 5.5.1. Clock Hardware 328
 - 5.5.2. Clock Software 329
 - 5.5.3. Soft Timers 332
- 5.6. CHARACTER-ORIENTED TERMINALS 333
 - 5.6.1. RS-232 Terminal Hardware 334
 - 5.6.2. Input Software 336
 - 5.6.3. Output Software 341
- 5.7. GRAPHICAL USER INTERFACES 342
 - 5.7.1. Personal Computer Keyboard, Mouse, and Display Hardware 343
 - 5.7.2. Input Software 347
 - 5.7.3. Output Software for Windows 347
- 5.8. NETWORK TERMINALS 355
 - 5.8.1. The X Window System 356
 - 5.8.2. The SLIM Network Terminal 360
- 5.9. POWER MANAGEMENT 363
 - 5.9.1. Hardware Issues 364
 - 5.9.2. Operating System Issues 365
 - 5.9.3. Degraded Operation 370
- 5.10. RESEARCH ON INPUT/OUTPUT 371
- 5.11. SUMMARY 372

6 FILE SYSTEMS**379**

- 6.1. FILES 380
 - 6.1.1. File Naming 380
 - 6.1.2. File Structure 382
 - 6.1.3. File Types 383
 - 6.1.4. File Access 385
 - 6.1.5. File Attributes 386
 - 6.1.6. File Operations 387
 - 6.1.7. An Example Program Using File System Calls 389
 - 6.1.8. Memory-Mapped Files 391

- 6.2. DIRECTORIES 393
 - 6.2.1. Single-Level Directory Systems 393
 - 6.2.2. Two-level Directory Systems 394
 - 6.2.3. Hierarchical Directory Systems 395
 - 6.2.4. Path Names 395
 - 6.2.5. Directory Operations 398

- 6.3. FILE SYSTEM IMPLEMENTATION 399
 - 6.3.1. File System Layout 399
 - 6.3.2. Implementing Files 400
 - 6.3.3. Implementing Directories 405
 - 6.3.4. Shared Files 408
 - 6.3.5. Disk Space Management 410
 - 6.3.6. File System Reliability 416
 - 6.3.7. File System Performance 424
 - 6.3.8. Log-Structured File Systems 428

- 6.4. EXAMPLE FILE SYSTEMS 430
 - 6.4.1. CD-ROM File Systems 430
 - 6.4.2. The CP/M File System 435
 - 6.4.3. The MS-DOS File System 438
 - 6.4.4. The Windows 98 File System 442
 - 6.4.5. The UNIX V7 File System 445

- 6.5. RESEARCH ON FILE SYSTEMS 448

- 6.6. SUMMARY 448

7 MULTIMEDIA OPERATING SYSTEMS**453**

- 7.1. INTRODUCTION TO MULTIMEDIA 454
- 7.2. MULTIMEDIA FILES 458
 - 7.2.1. Audio Encoding 459
 - 7.2.2. Video Encoding 461
- 7.3. VIDEO COMPRESSION 463
 - 7.3.1. The JPEG Standard 464
 - 7.3.2. The MPEG Standard 467
- 7.4. MULTIMEDIA PROCESS SCHEDULING 469
 - 7.4.1. Scheduling Homogeneous Processes 469
 - 7.4.2. General Real-Time Scheduling 470
 - 7.4.3. Rate Monotonic Scheduling 472
 - 7.4.4. Earliest Deadline First Scheduling 473
- 7.5. MULTIMEDIA FILE SYSTEM PARADIGMS 475
 - 7.5.1. VCR Control Functions 476
 - 7.5.2. Near Video on Demand 478
 - 7.5.3. Near Video on Demand with VCR Functions 479
- 7.6. FILE PLACEMENT 481
 - 7.6.1. Placing a File on a Single Disk 481
 - 7.6.2. Two Alternative File Organization Strategies 482
 - 7.6.3. Placing Files for Near Video on Demand 486
 - 7.6.4. Placing Multiple Files on a Single Disk 487
 - 7.6.5. Placing Files on Multiple Disks 490
- 7.7. CACHING 492
 - 7.7.1. Block Caching 492
 - 7.7.2. File Caching 494
- 7.8. DISK SCHEDULING FOR MULTIMEDIA 494
 - 7.8.1. Static Disk Scheduling 495
 - 7.8.2. Dynamic Disk Scheduling 496
- 7.9. RESEARCH ON MULTIMEDIA 498
- 7.10. SUMMARY 499

8 MULTIPLE PROCESSOR SYSTEMS 503

- 8.1. MULTIPROCESSORS 506
 - 8.1.1. Multiprocessor Hardware 506
 - 8.1.2. Multiprocessor Operating System Types 513
 - 8.1.3. Multiprocessor Synchronization 516
 - 8.1.4. Multiprocessor Scheduling 521

- 8.2. MULTICOMPUTERS 526
 - 8.2.1. Multicomputer Hardware 527
 - 8.2.2. Low-Level Communication Software 531
 - 8.2.3. User-Level Communication Software 534
 - 8.2.4. Remote Procedure Call 537
 - 8.2.5. Distributed Shared Memory 540
 - 8.2.6. Multicomputer Scheduling 544
 - 8.2.7. Load Balancing 545

- 8.3. DISTRIBUTED SYSTEMS 549
 - 8.3.1. Network Hardware 551
 - 8.3.2. Network Services and Protocols 553
 - 8.3.3. Document-Based Middleware 558
 - 8.3.4. File System-Based Middleware 559
 - 8.3.5. Shared Object-Based Middleware 565
 - 8.3.6. Coordination-Based Middleware 572

- 8.4. RESEARCH ON MULTIPLE PROCESSOR SYSTEMS 577

- 8.5. SUMMARY 577

9 SECURITY 583

- 9.1. THE SECURITY ENVIRONMENT 584
 - 9.1.1. Threats 584
 - 9.1.2. Intruders 585
 - 9.1.3. Accidental Data Loss 586

- 9.2. BASICS OF CRYPTOGRAPHY 587
 - 9.2.1. Secret-Key Cryptography 588
 - 9.2.2. Public-Key Cryptography 588

- 9.2.3. One-Way Functions 589
- 9.2.4. Digital Signatures 590
- 9.3. USER AUTHENTICATION 591
 - 9.3.1. Authentication Using Passwords 592
 - 9.3.2. Authentication Using a Physical Object 601
 - 9.3.3. Authentication Using Biometrics 603
 - 9.3.4. Countermeasures 606
- 9.4. ATTACKS FROM INSIDE THE SYSTEM 606
 - 9.4.1. Trojan Horses 607
 - 9.4.2. Login Spoofing 608
 - 9.4.3. Logic Bombs 609
 - 9.4.4. Trap Doors 610
 - 9.4.5. Buffer Overflow 610
 - 9.4.6. Generic Security Attacks 613
 - 9.4.7. Famous Security Flaws 614
 - 9.4.8. Design Principles for Security 616
- 9.5. ATTACKS FROM OUTSIDE THE SYSTEM 617
 - 9.5.1. Virus Damage Scenarios 618
 - 9.5.2. How Viruses Work 619
 - 9.5.3. How Viruses Spread 626
 - 9.5.4. Antivirus and Anti-Antivirus Techniques 628
 - 9.5.5. The Internet Worm 635
 - 9.5.6. Mobile Code 637
 - 9.5.7. Java Security 642
- 9.6. PROTECTION MECHANISMS 645
 - 9.6.1. Protection Domains 645
 - 9.6.2. Access Control Lists 647
 - 9.6.3. Capabilities 650
- 9.7. TRUSTED SYSTEMS 653
 - 9.7.1. Trusted Computing Base 654
 - 9.7.2. Formal Models of Secure Systems 655
 - 9.7.3. Multilevel Security 657
 - 9.7.4. Orange Book Security 659
 - 9.7.5. Covert Channels 661
- 9.8. RESEARCH ON SECURITY 665
- 9.9. SUMMARY 666

10 CASE STUDY 1: UNIX AND LINUX**671**

- 10.1. HISTORY OF UNIX 672
 - 10.1.1. UNICS 672
 - 10.1.2. PDP-11 UNIX 673
 - 10.1.3. Portable UNIX 674
 - 10.1.4. Berkeley UNIX 675
 - 10.1.5. Standard UNIX 676
 - 10.1.6. MINIX 677
 - 10.1.7. Linux 678

- 10.2. OVERVIEW OF UNIX 681
 - 10.2.1. UNIX Goals 681
 - 10.2.2. Interfaces to UNIX 682
 - 10.2.3. The UNIX Shell 683
 - 10.2.4. UNIX Utility Programs 686
 - 10.2.5. Kernel Structure 687

- 10.3. PROCESSES IN UNIX 690
 - 10.3.1. Fundamental Concepts 690
 - 10.3.2. Process Management System Calls in UNIX 692
 - 10.3.3. Implementation of Processes in UNIX 699
 - 10.3.4. Booting UNIX 708

- 10.4. MEMORY MANAGEMENT IN UNIX 710
 - 10.4.1. Fundamental Concepts 711
 - 10.4.2. Memory Management System Calls in UNIX 714
 - 10.4.3. Implementation of Memory Management in UNIX 715

- 10.5. INPUT/OUTPUT IN UNIX 723
 - 10.5.1. Fundamental Concepts 724
 - 10.5.2. Input/Output System Calls in UNIX 726
 - 10.5.3. Implementation of Input/Output in UNIX 727
 - 10.5.4. Streams 730

- 10.6. THE UNIX FILE SYSTEM 732
 - 10.6.1. Fundamental Concepts 732
 - 10.6.2. File System Calls in UNIX 736
 - 10.6.3. Implementation of the UNIX File System 740
 - 10.6.4. NFS: The Network File System 747

- 10.7. SECURITY IN UNIX 753
 - 10.7.1. Fundamental Concepts 753
 - 10.7.2. Security System Calls in UNIX 755
 - 10.7.3. Implementation of Security in UNIX 756
- 10.8. SUMMARY 757

11 CASE STUDY 2: WINDOWS 2000

763

- 11.1. HISTORY OF WINDOWS 2000 763
 - 11.1.1. MS-DOS 763
 - 11.1.2. Windows 95/98/Me 764
 - 11.1.3. Windows NT 765
 - 11.1.4. Windows 2000 767
- 11.2. PROGRAMMING WINDOWS 2000 771
 - 11.2.1. The Win32 Application Programming Interface 772
 - 11.2.2. The Registry 774
- 11.3. SYSTEM STRUCTURE 778
 - 11.3.1. Operating System Structure 778
 - 11.3.2. Implementation of Objects 787
 - 11.3.3. Environment Subsystems 792
- 11.4. PROCESSES AND THREADS IN WINDOWS 2000 796
 - 11.4.1. Fundamental Concepts 796
 - 11.4.2. Job, Process, Thread and Fiber Management API Calls 799
 - 11.4.3. Implementation of Processes and Threads 802
 - 11.4.4. MS-DOS Emulation 809
 - 11.4.5. Booting Windows 2000 820
- 11.5. MEMORY MANAGEMENT 811
 - 11.5.1. Fundamental Concepts 812
 - 11.5.2. Memory Management System Calls 816
 - 11.5.3. Implementation of Memory Management 817
- 11.6. INPUT/OUTPUT IN WINDOWS 2000 824
 - 11.6.1. Fundamental Concepts 824
 - 11.6.2. Input/Output API Calls 825
 - 11.6.3. Implementation of I/O 827
 - 11.6.4. Device Drivers 827

- 11.7. THE WINDOWS 2000 FILE SYSTEM 830
 - 11.7.1. Fundamental Concepts 830
 - 11.7.2. File System API Calls in Windows 2000 831
 - 11.7.3. Implementation of the Windows 2000 File System 833
- 11.8. SECURITY IN WINDOWS 2000 844
 - 11.8.1. Fundamental Concepts 845
 - 11.8.2. Security API Calls 847
 - 11.8.3. Implementation of Security 848
- 11.9. CACHING IN WINDOWS 2000 849
- 11.10. SUMMARY 851

12 OPERATING SYSTEM DESIGN

855

- 12.1. THE NATURE OF THE DESIGN PROBLEM 856
 - 12.1.1. Goals 856
 - 12.1.2. Why is it Hard to Design an Operating Systems? 857
- 12.2. INTERFACE DESIGN 859
 - 12.2.1. Guiding Principles 859
 - 12.2.2. Paradigms 861
 - 12.2.3. The System Call Interface 864
- 12.3. IMPLEMENTATION 867
 - 12.3.1. System Structure 867
 - 12.3.2. Mechanism versus Policy 870
 - 12.3.3. Orthogonality 871
 - 12.3.4. Naming 872
 - 12.3.5. Binding Time 874
 - 12.3.6. Static versus Dynamic Structures 875
 - 12.3.7. Top-Down versus Bottom-Up Implementation 876
 - 12.3.8. Useful Techniques 877
- 12.4. PERFORMANCE 882
 - 12.4.1. Why are Operating Systems Slow? 882
 - 12.4.2. What Should be Optimized? 883
 - 12.4.3. Space-Time Trade-offs 884
 - 12.4.4. Caching 887

- 12.4.5. Hints 888
- 12.4.6. Exploiting Locality 888
- 12.4.7. Optimize the Common Case 889
- 12.5. PROJECT MANAGEMENT 889
 - 12.5.1. The Mythical Man Month 890
 - 12.5.2. Team Structure 891
 - 12.5.3. The Role of Experience 893
 - 12.5.4. No Silver Bullet 894
- 12.6. TRENDS IN OPERATING SYSTEM DESIGN 894
 - 12.6.1. Large Address Space Operating Systems 894
 - 12.6.2. Networking 895
 - 12.6.3. Parallel and Distributed Systems 896
 - 12.6.4. Multimedia 896
 - 12.6.5. Battery-Powered Computers 896
 - 12.6.6. Embedded Systems 897
- 12.7. SUMMARY 897

13 READING LIST AND BIBLIOGRAPHY

901

- 13.1. SUGGESTIONS FOR FURTHER READING 901
 - 13.1.1. Introduction and General Works 902
 - 13.1.2. Processes and Threads 902
 - 13.1.3. Deadlocks 903
 - 13.1.4. Memory Management 903
 - 13.1.5. Input/Output 903
 - 13.1.6. File Systems 904
 - 13.1.7. Multimedia Operating Systems 905
 - 13.1.8. Multiple Processor Systems 906
 - 13.1.9. Security 907
 - 13.1.10. UNIX and Linux 908
 - 13.1.11. Windows 2000 909
 - 13.1.12. Design Principles 910
- 13.2. ALPHABETICAL BIBLIOGRAPHY 911

INDEX

935

PREFACE

The world has changed a great deal since the first edition of this book appeared in 1992. Computer networks and distributed systems of all kinds have become very common. Small children now roam the Internet, where previously only computer professionals went. As a consequence, this book has changed a great deal, too.

The most obvious change is that the first edition was about half on single-processor operating systems and half on distributed systems. I chose that format in 1991 because few universities then had courses on distributed systems and whatever students learned about distributed systems had to be put into the operating systems course, for which this book was intended. Now most universities have a separate course on distributed systems, so it is not necessary to try to combine the two subjects into one course and one book. This book is intended for a first course on operating systems, and as such focuses mostly on traditional single-processor systems.

I have coauthored two other books on operating systems. This leads to two possible course sequences.

Practically-oriented sequence:

1. Operating Systems Design and Implementation by Tanenbaum and Woodhull
2. Distributed Systems by Tanenbaum and Van Steen

Traditional sequence:

1. Modern Operating Systems by Tanenbaum
2. Distributed Systems by Tanenbaum and Van Steen

The former sequence uses MINIX and the students are expected to experiment with MINIX in an accompanying laboratory supplementing the first course. The latter sequence does not use MINIX. Instead, some small simulators are available that can be used for student exercises during a first course using this book. These simulators can be found starting on the author's Web page: www.cs.vu.nl/~ast/ by clicking on Software and supplementary material for my books.

In addition to the major change of switching the emphasis to single-processor operating systems in this book, other major changes include the addition of entire chapters on computer security, multimedia operating systems, and Windows 2000, all important and timely topics. In addition, a new and unique chapter on operating system design has been added.

Another new feature is that many chapters now have a section on research about the topic of the chapter. This is intended to introduce the reader to modern work in processes, memory management, and so on. These sections have numerous references to the current research literature for the interested reader. In addition, Chapter 13 has many introductory and tutorial references.

Finally, numerous topics have been added to this book or heavily revised. These topics include: graphical user interfaces, multiprocessor operating systems, power management for laptops, trusted systems, viruses, network terminals, CD-ROM file systems, mutexes, RAID, soft timers, stable storage, fair-share scheduling, and new paging algorithms. Many new problems have been added and old ones updated. The total number of problems now exceeds 450. A solutions manual is available to professors using this book in a course. They can obtain a copy from their local Prentice Hall representative. In addition, over 250 new references to the current literature have been added to bring the book up to date.

Despite the removal of more than 400 pages of old material, the book has increased in size due to the large amount of new material added. While the book is still suitable for a one-semester or two-quarter course, it is probably too long for a one-quarter or one-trimester course at most universities. For this reason, the book has been designed in a modular way. Any course on operating systems should cover chapters 1 through 6. This is basic material that every student should know.

If additional time is available, additional chapters can be covered. Each of them assumes the reader has finished chapters 1 through 6, but Chaps. 7 through 12 are each self contained, so any desired subset can be used and in any order, depending on the interests of the instructor. In the author's opinion, Chaps. 7 through 12 are much more interesting than the earlier ones. Instructors should tell their students that they have to eat their broccoli before they can have the double chocolate fudge cake dessert.

I would like to thank the following people for their help in reviewing parts of the manuscript: Rida Bazzi, Riccardo Bettati, Felipe Cabrera, Richard Chapman, John Connely, John Dickinson, John Elliott, Deborah Frincke, Chandana Gamage, Robbert Geist, David Golds, Jim Griffioen, Gary Harkin, Frans Kaashoek, Muk-

kai Krishnamoorthy, Monica Lam, Jussi Leiwo, Herb Mayer, Kirk McKusick, Evi Nemeth, Bill Potvin, Prasant Shenoy, Thomas Skinner, Xian-He Sun, William Terry, Robbert Van Renesse, and Maarten van Steen. Jamie Hanrahan, Mark Russinovich, and Dave Solomon were enormously knowledgeable about Windows 2000 and very helpful. Special thanks go to Al Woodhull for valuable reviews and thinking of many new end-of-chapter problems.

My students were also helpful with comments and feedback, especially Staas de Jong, Jan de Vos, Niels Drost, David Fokkema, Auke Folkerts, Peter Groenewegen, Wilco Ibes, Stefan Jansen, Jeroen Ketema, Joeri Mulder, Irwin Oppenheim, Stef Post, Umar Rehman, Daniel Rijkhof, Maarten Sander, Maurits van der Schee, Rik van der Stoel, Mark van Driel, Dennis van Veen, and Thomas Zeeman.

Barbara and Marvin are still wonderful, as usual, each in a unique way. Finally, last but not least, I would like to thank Suzanne for her love and patience, not to mention all the *druiven* and *kersen*, which have replaced the *sinasappelsap* in recent times.

Andrew S. Tanenbaum