

MODERN OPERATING SYSTEMS

Fifth Edition

Andrew S.
Tanenbaum
Herbert
Bos



**MODERN
OPERATING SYSTEMS**

FIFTH EDITION

This page intentionally left blank

**MODERN
OPERATING SYSTEMS**

FIFTH EDITION

**ANDREW S. TANENBAUM
HERBERT BOS**

*Vrije Universiteit
Amsterdam, The Netherlands*



Content Management: Tracy Johnson, Erin Sullivan

Content Production: Carole Snyder, Pallavi Pandit

Product Management: Holly Stark

Product Marketing: Krista Clark

Rights and Permissions: Anjali Singh

Please contact <https://support.pearson.com/getsupport/s/> with any queries on this content.

Cover Image by Jason Consalvo.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Copyright © 2023, 2014, 2008 by Pearson Education, Inc. or its affiliates, 221 River Street, Hoboken, NJ 07030. All Rights Reserved. Manufactured in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights and Permissions department, please visit www.pearsoned.com/permissions/.

Acknowledgments of third-party content appear on the appropriate page within the text.

PEARSON is an exclusive trademark owned by Pearson Education, Inc. or its affiliates in the U.S. and/or other countries.

Unless otherwise indicated herein, any third-party trademarks, logos, or icons that may appear in this work are the property of their respective owners, and any references to third-party trademarks, logos, icons, or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson’s products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc., or its affiliates, authors, licensees, or distributors.

Library of Congress Cataloging-in-Publication Data

MODERN OPERATING SYSTEMS

Library of Congress Control Number: 2022902564

ScoutAutomatedPrintCode

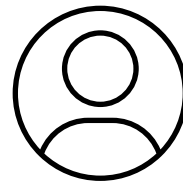


Rental
ISBN-10: 0-13-761887-5
ISBN-13: 978-0-13-761887-3

To Suzanne, Barbara, Aron, Nathan, Marvin, Matilde, Olivia and Mirte. (AST)

To Marieke, Duko, Jip, Loeka and Spot. (HB)

Pearson's Commitment to Diversity, Equity, and Inclusion



Pearson is dedicated to creating bias-free content that reflects the diversity, depth, and breadth of all learners' lived experiences.

We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, sex, sexual orientation, socioeconomic status, ability, age, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational content accurately reflects the histories and lived experiences of the learners we serve.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content prompts deeper discussions with students and motivates them to expand their own learning (and worldview).

Accessibility

We are also committed to providing products that are fully accessible to all learners. As per Pearson's guidelines for accessible educational Web media, we test and retest the capabilities of our products against the highest standards for every release, following the WCAG guidelines in developing new products for copyright year 2022 and beyond.

 You can learn more about Pearson's commitment to accessibility at <https://www.pearson.com/us/accessibility.html>

Contact Us

While we work hard to present unbiased, fully accessible content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.



Please contact us with concerns about any potential bias at <https://www.pearson.com/report-bias.html>



For accessibility-related issues, such as using assistive technology with Pearson products, alternative text requests, or accessibility documentation, email the Pearson Disability Support team at disability.support@pearson.com



CONTENTS

PREFACE

xxiii

1 INTRODUCTION

1

- 1.1 WHAT IS AN OPERATING SYSTEM? 4
 - 1.1.1 The Operating System as an Extended Machine 4
 - 1.1.2 The Operating System as a Resource Manager 6
- 1.2 HISTORY OF OPERATING SYSTEMS 7
 - 1.2.1 The First Generation (1945–1955): Vacuum Tubes 8
 - 1.2.2 The Second Generation (1955–1965): Transistors and Batch Systems 8
 - 1.2.3 The Third Generation (1965–1980): ICs and Multiprogramming 10
 - 1.2.4 The Fourth Generation (1980–Present): Personal Computers 15
 - 1.2.5 The Fifth Generation (1990–Present): Mobile Computers 19
- 1.3 COMPUTER HARDWARE REVIEW 20
 - 1.3.1 Processors 20
 - 1.3.2 Memory 24
 - 1.3.3 Nonvolatile Storage 28
 - 1.3.4 I/O Devices 29
 - 1.3.5 Buses 33
 - 1.3.6 Booting the Computer 34

- 1.4 THE OPERATING SYSTEM ZOO 36
 - 1.4.1 Mainframe Operating Systems 36
 - 1.4.2 Server Operating Systems 37
 - 1.4.3 Personal Computer Operating Systems 37
 - 1.4.4 Smartphone and Handheld Computer Operating Systems 37
 - 1.4.5 The Internet of Things and Embedded Operating Systems 37
 - 1.4.6 Real-Time Operating Systems 38
 - 1.4.7 Smart Card Operating Systems 39
- 1.5 OPERATING SYSTEM CONCEPTS 39
 - 1.5.1 Processes 39
 - 1.5.2 Address Spaces 41
 - 1.5.3 Files 42
 - 1.5.4 Input/Output 45
 - 1.5.5 Protection 46
 - 1.5.6 The Shell 46
 - 1.5.7 Ontogeny Recapitulates Phylogeny 47
- 1.6 SYSTEM CALLS 50
 - 1.6.1 System Calls for Process Management 53
 - 1.6.2 System Calls for File Management 57
 - 1.6.3 System Calls for Directory Management 58
 - 1.6.4 Miscellaneous System Calls 60
 - 1.6.5 The Windows API 60
- 1.7 OPERATING SYSTEM STRUCTURE 63
 - 1.7.1 Monolithic Systems 63
 - 1.7.2 Layered Systems 64
 - 1.7.3 Microkernels 66
 - 1.7.4 Client-Server Model 68
 - 1.7.5 Virtual Machines 69
 - 1.7.6 Exokernels and Unikernels 73
- 1.8 THE WORLD ACCORDING TO C 74
 - 1.8.1 The C Language 74
 - 1.8.2 Header Files 75
 - 1.8.3 Large Programming Projects 76
 - 1.8.4 The Model of Run Time 78
- 1.9 RESEARCH ON OPERATING SYSTEMS 78
- 1.10 OUTLINE OF THE REST OF THIS BOOK 79

1.11 METRIC UNITS 80

1.12 SUMMARY 81

2 PROCESSES AND THREADS

85

2.1 PROCESSES 85

2.1.1 The Process Model 86

2.1.2 Process Creation 88

2.1.3 Process Termination 90

2.1.4 Process Hierarchies 91

2.1.5 Process States 92

2.1.6 Implementation of Processes 94

2.1.7 Modeling Multiprogramming 95

2.2 THREADS 97

2.2.1 Thread Usage 97

2.2.2 The Classical Thread Model 102

2.2.3 POSIX Threads 106

2.2.4 Implementing Threads in User Space 107

2.2.5 Implementing Threads in the Kernel 111

2.2.6 Hybrid Implementations 112

2.2.7 Making Single-Threaded Code Multithreaded 113

2.3 EVENT-DRIVEN SERVERS 116

2.4 SYNCHRONIZATION AND INTERPROCESS COMMUNICATION 119

2.4.1 Race Conditions 119

2.4.2 Critical Regions 120

2.4.3 Mutual Exclusion with Busy Waiting 121

2.4.4 Sleep and Wakeup 127

2.4.5 Semaphores 129

2.4.6 Mutexes 134

2.4.7 Monitors 138

2.4.8 Message Passing 145

2.4.9 Barriers 148

2.4.10 Priority Inversion 150

2.4.11 Avoiding Locks: Read-Copy-Update 151

- 2.5 SCHEDULING 152
 - 2.5.1 Introduction to Scheduling 153
 - 2.5.2 Scheduling in Batch Systems 160
 - 2.5.3 Scheduling in Interactive Systems 162
 - 2.5.4 Scheduling in Real-Time Systems 168
 - 2.5.5 Policy Versus Mechanism 169
 - 2.5.6 Thread Scheduling 169
- 2.6 RESEARCH ON PROCESSES AND THREADS 171
- 2.7 SUMMARY 172

3 MEMORY MANAGEMENT 179

- 3.1 NO MEMORY ABSTRACTION 180
 - 3.1.1 Running Multiple Programs Without a Memory Abstraction 181
- 3.2 A MEMORY ABSTRACTION: ADDRESS SPACES 183
 - 3.2.1 The Notion of an Address Space 184
 - 3.2.2 Swapping 185
 - 3.2.3 Managing Free Memory 188
- 3.3 VIRTUAL MEMORY 192
 - 3.3.1 Paging 193
 - 3.3.2 Page Tables 196
 - 3.3.3 Speeding Up Paging 200
 - 3.3.4 Page Tables for Large Memories 203
- 3.4 PAGE REPLACEMENT ALGORITHMS 207
 - 3.4.1 The Optimal Page Replacement Algorithm 208
 - 3.4.2 The Not Recently Used Page Replacement Algorithm 209
 - 3.4.3 The First-In, First-Out (FIFO) Page Replacement Algorithm 210
 - 3.4.4 The Second-Chance Page Replacement Algorithm 210
 - 3.4.5 The Clock Page Replacement Algorithm 211
 - 3.4.6 The Least Recently Used (LRU) Page Replacement Algorithm 212
 - 3.4.7 Simulating LRU in Software 212
 - 3.4.8 The Working Set Page Replacement Algorithm 214
 - 3.4.9 The WSClock Page Replacement Algorithm 218
 - 3.4.10 Summary of Page Replacement Algorithms 220

- 3.5 DESIGN ISSUES FOR PAGING SYSTEMS 221
 - 3.5.1 Local versus Global Allocation Policies 221
 - 3.5.2 Load Control 224
 - 3.5.3 Cleaning Policy 225
 - 3.5.4 Page Size 226
 - 3.5.5 Separate Instruction and Data Spaces 227
 - 3.5.6 Shared Pages 228
 - 3.5.7 Shared Libraries 230
 - 3.5.8 Mapped Files 232
- 3.6 IMPLEMENTATION ISSUES 232
 - 3.6.1 Operating System Involvement with Paging 232
 - 3.6.2 Page Fault Handling 233
 - 3.6.3 Instruction Backup 234
 - 3.6.4 Locking Pages in Memory 236
 - 3.6.5 Backing Store 236
 - 3.6.6 Separation of Policy and Mechanism 238
- 3.7 SEGMENTATION 240
 - 3.7.1 Implementation of Pure Segmentation 242
 - 3.7.2 Segmentation with Paging: MULTICS 243
 - 3.7.3 Segmentation with Paging: The Intel x86 248
- 3.8 RESEARCH ON MEMORY MANAGEMENT 248
- 3.9 SUMMARY 250

4 FILE SYSTEMS

259

- 4.1 FILES 261
 - 4.1.1 File Naming 261
 - 4.1.2 File Structure 263
 - 4.1.3 File Types 264
 - 4.1.4 File Access 266
 - 4.1.5 File Attributes 267
 - 4.1.6 File Operations 269
 - 4.1.7 An Example Program Using File-System Calls 270

- 4.2 DIRECTORIES 272
 - 4.2.1 Single-Level Directory Systems 272
 - 4.2.2 Hierarchical Directory Systems 273
 - 4.2.3 Path Names 274
 - 4.2.4 Directory Operations 277

- 4.3 FILE-SYSTEM IMPLEMENTATION 278
 - 4.3.1 File-System Layout 278
 - 4.3.2 Implementing Files 280
 - 4.3.3 Implementing Directories 285
 - 4.3.4 Shared Files 288
 - 4.3.5 Log-Structured File Systems 290
 - 4.3.6 Journaling File Systems 292
 - 4.3.7 Flash-based File Systems 293
 - 4.3.8 Virtual File Systems 298

- 4.4 FILE-SYSTEM MANAGEMENT AND OPTIMIZATION 301
 - 4.4.1 Disk-Space Management 301
 - 4.4.2 File-System Backups 307
 - 4.4.3 File-System Consistency 312
 - 4.4.4 File-System Performance 315
 - 4.4.5 Defragmenting Disks 320
 - 4.4.6 Compression and Deduplication 321
 - 4.4.7 Secure File Deletion and Disk Encryption 322

- 4.5 EXAMPLE FILE SYSTEMS 324
 - 4.5.1 The MS-DOS File System 324
 - 4.5.2 The UNIX V7 File System 327

- 4.6 RESEARCH ON FILE SYSTEMS 330

- 4.7 SUMMARY 331

5 INPUT/OUTPUT

337

- 5.1 PRINCIPLES OF I/O HARDWARE 337
 - 5.1.1 I/O Devices 338
 - 5.1.2 Device Controllers 338
 - 5.1.3 Memory-Mapped I/O 340

- 5.1.4 Direct Memory Access 344
- 5.1.5 Interrupts Revisited 347

- 5.2 PRINCIPLES OF I/O SOFTWARE 352
 - 5.2.1 Goals of the I/O Software 352
 - 5.2.2 Programmed I/O 354
 - 5.2.3 Interrupt-Driven I/O 355
 - 5.2.4 I/O Using DMA 356

- 5.3 I/O SOFTWARE LAYERS 357
 - 5.3.1 Interrupt Handlers 357
 - 5.3.2 Device Drivers 359
 - 5.3.3 Device-Independent I/O Software 362
 - 5.3.4 User-Space I/O Software 368

- 5.4 MASS STORAGE: DISK AND SSD 370
 - 5.4.1 Magnetic Disks 370
 - 5.4.2 Solid State Drives (SSDs) 381
 - 5.4.3 RAID 385

- 5.5 CLOCKS 390
 - 5.5.1 Clock Hardware 390
 - 5.5.2 Clock Software 391
 - 5.5.3 Soft Timers 394

- 5.6 USER INTERFACES: KEYBOARD, MOUSE, & MONITOR 395
 - 5.6.1 Input Software 396
 - 5.6.2 Output Software 402

- 5.7 THIN CLIENTS 419

- 5.8 POWER MANAGEMENT 420
 - 5.8.1 Hardware Issues 421
 - 5.8.2 Operating System Issues 422
 - 5.8.3 Application Program Issues 428

- 5.9 RESEARCH ON INPUT/OUTPUT 428

- 5.10 SUMMARY 430

6 DEADLOCKS 437

- 6.1 RESOURCES 438
 - 6.1.1 Preemptable and Nonpreemptable Resources 438
 - 6.1.2 Resource Acquisition 439
 - 6.1.3 The Dining Philosophers Problem 440
- 6.2 INTRODUCTION TO DEADLOCKS 444
 - 6.2.1 Conditions for Resource Deadlocks 445
 - 6.2.2 Deadlock Modeling 445
- 6.3 THE OSTRICH ALGORITHM 447
- 6.4 DEADLOCK DETECTION AND RECOVERY 449
 - 6.4.1 Deadlock Detection with One Resource of Each Type 449
 - 6.4.2 Deadlock Detection with Multiple Resources of Each Type 451
 - 6.4.3 Recovery from Deadlock 454
- 6.5 DEADLOCK AVOIDANCE 455
 - 6.5.1 Resource Trajectories 456
 - 6.5.2 Safe and Unsafe States 457
 - 6.5.3 The Banker's Algorithm for a Single Resource 458
 - 6.5.4 The Banker's Algorithm for Multiple Resources 459
- 6.6 DEADLOCK PREVENTION 461
 - 6.6.1 Attacking the Mutual-Exclusion Condition 461
 - 6.6.2 Attacking the Hold-and-Wait Condition 462
 - 6.6.3 Attacking the No-Preemption Condition 462
 - 6.6.4 Attacking the Circular Wait Condition 463
- 6.7 OTHER ISSUES 464
 - 6.7.1 Two-Phase Locking 464
 - 6.7.2 Communication Deadlocks 465
 - 6.7.3 Livelock 467
 - 6.7.4 Starvation 468
- 6.8 RESEARCH ON DEADLOCKS 469
- 6.9 SUMMARY 470

7 VIRTUALIZATION AND THE CLOUD 477

- 7.1 HISTORY 480
- 7.2 REQUIREMENTS FOR VIRTUALIZATION 482
- 7.3 TYPE 1 AND TYPE 2 HYPERVISORS 484
- 7.4 TECHNIQUES FOR EFFICIENT VIRTUALIZATION 486
 - 7.4.1 Virtualizing the Unvirtualizable 487
 - 7.4.2 The Cost of Virtualization 489
- 7.5 ARE HYPERVISORS MICROKERNELS DONE RIGHT? 490
- 7.6 MEMORY VIRTUALIZATION 493
- 7.7 I/O VIRTUALIZATION 497
- 7.8 VIRTUAL MACHINES ON MULTICORE CPUS 501
- 7.9 CLOUDS 501
 - 7.9.1 Clouds as a Service 502
 - 7.9.2 Virtual Machine Migration 503
 - 7.9.3 Checkpointing 504
- 7.10 OS-LEVEL VIRTUALIZATION 504
- 7.11 CASE STUDY: VMWARE 507
 - 7.11.1 The Early History of VMware 507
 - 7.11.2 VMware Workstation 509
 - 7.11.3 Challenges in Bringing Virtualization to the x86 509
 - 7.11.4 VMware Workstation: Solution Overview 511
 - 7.11.5 The Evolution of VMware Workstation 520
 - 7.11.6 ESX Server: VMware's type 1 Hypervisor 521
- 7.12 RESEARCH ON VIRTUALIZATION AND THE CLOUD 523
- 7.13 SUMMARY 524

8 MULTIPLE PROCESSOR SYSTEMS 527

- 8.1 MULTIPROCESSORS 530
 - 8.1.1 Multiprocessor Hardware 530
 - 8.1.2 Multiprocessor Operating System Types 541
 - 8.1.3 Multiprocessor Synchronization 545
 - 8.1.4 Multiprocessor Scheduling 550
- 8.2 MULTICOMPUTERS 557
 - 8.2.1 Multicomputer Hardware 558
 - 8.2.2 Low-Level Communication Software 562
 - 8.2.3 User-Level Communication Software 565
 - 8.2.4 Remote Procedure Call 569
 - 8.2.5 Distributed Shared Memory 571
 - 8.2.6 Multicomputer Scheduling 575
 - 8.2.7 Load Balancing 576
- 8.3 DISTRIBUTED SYSTEMS 579
 - 8.3.1 Network Hardware 581
 - 8.3.2 Network Services and Protocols 585
 - 8.3.3 Document-Based Middleware 588
 - 8.3.4 File-System-Based Middleware 590
 - 8.3.5 Object-Based Middleware 594
 - 8.3.6 Coordination-Based Middleware 595
- 8.4 RESEARCH ON MULTIPLE PROCESSOR SYSTEMS 598
- 8.5 SUMMARY 600

9 SECURITY 605

- 9.1 FUNDAMENTALS OF OPERATING SYSTEM SECURITY 607
 - 9.1.1 The CIA Security Triad 608
 - 9.1.2 Security Principles 609
 - 9.1.3 Security of the Operating System Structure 611
 - 9.1.4 Trusted Computing Base 612
 - 9.1.5 Attackers 614
 - 9.1.6 Can We Build Secure Systems? 617

- 9.2 CONTROLLING ACCESS TO RESOURCES 618
 - 9.2.1 Protection Domains 619
 - 9.2.2 Access Control Lists 621
 - 9.2.3 Capabilities 625
- 9.3 FORMAL MODELS OF SECURE SYSTEMS 628
 - 9.3.1 Multilevel Security 629
 - 9.3.2 Cryptography 632
 - 9.3.3 Trusted Platform Modules 636
- 9.4 AUTHENTICATION 637
 - 9.4.1 Passwords 637
 - 9.4.2 Authentication Using a Physical Object 644
 - 9.4.3 Authentication Using Biometrics 645
- 9.5 EXPLOITING SOFTWARE 647
 - 9.5.1 Buffer Overflow Attacks 648
 - 9.5.2 Format String Attacks 658
 - 9.5.3 Use-After-Free Attacks 661
 - 9.5.4 Type Confusion Vulnerabilities 662
 - 9.5.5 Null Pointer Dereference Attacks 664
 - 9.5.6 Integer Overflow Attacks 665
 - 9.5.7 Command Injection Attacks 666
 - 9.5.8 Time of Check to Time of Use Attacks 667
 - 9.5.9 Double Fetch Vulnerability 668
- 9.6 EXPLOITING HARDWARE 668
 - 9.6.1 Covert Channels 669
 - 9.6.2 Side Channels 671
 - 9.6.3 Transient Execution Attacks 674
- 9.7 INSIDER ATTACKS 679
 - 9.7.1 Logic Bombs 679
 - 9.7.2 Back Doors 680
 - 9.7.3 Login Spoofing 681
- 9.8 OPERATING SYSTEM HARDENING 681
 - 9.8.1 Fine-Grained Randomization 682
 - 9.8.2 Control-Flow Restrictions 683
 - 9.8.3 Access Restrictions 685
 - 9.8.4 Code and Data Integrity Checks 689
 - 9.8.5 Remote Attestation Using a Trusted Platform Module 690
 - 9.8.6 Encapsulating Untrusted Code 691

- 9.9 RESEARCH ON SECURITY 694
- 9.10 SUMMARY 696

10 CASE STUDY 1: UNIX,

703

- 10.1 HISTORY OF UNIX AND LINUX 704
 - 10.1.1 UNICS 704
 - 10.1.2 PDP-11 UNIX 705
 - 10.1.3 Portable UNIX 706
 - 10.1.4 Berkeley UNIX 707
 - 10.1.5 Standard UNIX 708
 - 10.1.6 MINIX 709
 - 10.1.7 Linux 710
- 10.2 OVERVIEW OF LINUX 713
 - 10.2.1 Linux Goals 713
 - 10.2.2 Interfaces to Linux 714
 - 10.2.3 The Shell 716
 - 10.2.4 Linux Utility Programs 719
 - 10.2.5 Kernel Structure 720
- 10.3 PROCESSES IN LINUX 723
 - 10.3.1 Fundamental Concepts 724
 - 10.3.2 Process-Management System Calls in Linux 726
 - 10.3.3 Implementation of Processes and Threads in Linux 730
 - 10.3.4 Scheduling in Linux 736
 - 10.3.5 Synchronization in Linux 740
 - 10.3.6 Booting Linux 741
- 10.4 MEMORY MANAGEMENT IN LINUX 743
 - 10.4.1 Fundamental Concepts 744
 - 10.4.2 Memory Management System Calls in Linux 746
 - 10.4.3 Implementation of Memory Management in Linux 748
 - 10.4.4 Paging in Linux 754
- 10.5 INPUT/OUTPUT IN LINUX 757
 - 10.5.1 Fundamental Concepts 758
 - 10.5.2 Networking 759

- 10.5.3 Input/Output System Calls in Linux 761
- 10.5.4 Implementation of Input/Output in Linux 762
- 10.5.5 Modules in Linux 765
- 10.6 THE LINUX FILE SYSTEM 766
 - 10.6.1 Fundamental Concepts 766
 - 10.6.2 File-System Calls in Linux 770
 - 10.6.3 Implementation of the Linux File System 774
 - 10.6.4 NFS: The Network File System 783
- 10.7 SECURITY IN LINUX 789
 - 10.7.1 Fundamental Concepts 789
 - 10.7.2 Security System Calls in Linux 791
 - 10.7.3 Implementation of Security in Linux 792
- 10.8 ANDROID 793
 - 10.8.1 Android and Google 794
 - 10.8.2 History of Android 794
 - 10.8.3 Design Goals 800
 - 10.8.4 Android Architecture 801
 - 10.8.5 Linux Extensions 803
 - 10.8.6 ART 807
 - 10.8.7 Binder IPC 809
 - 10.8.8 Android Applications 818
 - 10.8.9 Intents 830
 - 10.8.10 Process Model 831
 - 10.8.11 Security and Privacy 836
 - 10.8.12 Background Execution and Social Engineering 856
- 10.9 SUMMARY 863

11 CASE STUDY 2: WINDOWS 11

871

- 11.1 HISTORY OF WINDOWS THROUGH WINDOWS 11 871
 - 11.1.1 1980s: MS-DOS 872
 - 11.1.2 1990s: MS-DOS-based Windows 873
 - 11.1.3 2000s: NT-based Windows 873
 - 11.1.4 Windows Vista 876
 - 11.1.5 Windows 8 877

- 11.1.6 Windows 10 878
- 11.1.7 Windows 11 879

- 11.2 PROGRAMMING WINDOWS 880
 - 11.2.1 Universal Windows Platform 881
 - 11.2.2 Windows Subsystems 883
 - 11.2.3 The Native NT Application Programming Interface 884
 - 11.2.4 The Win32 Application Programming Interface 887
 - 11.2.5 The Windows Registry 891

- 11.3 SYSTEM STRUCTURE 894
 - 11.3.1 Operating System Structure 894
 - 11.3.2 Booting Windows 910
 - 11.3.3 Implementation of the Object Manager 914
 - 11.3.4 Subsystems, DLLs, and User-Mode Services 926

- 11.4 PROCESSES AND THREADS IN WINDOWS 929
 - 11.4.1 Fundamental Concepts 929
 - 11.4.2 Job, Process, Thread, and Fiber Management API Calls 934
 - 11.4.3 Implementation of Processes and Threads 941
 - 11.4.4 WoW64 and Emulation 950

- 11.5 MEMORY MANAGEMENT 955
 - 11.5.1 Fundamental Concepts 955
 - 11.5.2 Memory-Management System Calls 961
 - 11.5.3 Implementation of Memory Management 962
 - 11.5.4 Memory Compression 973
 - 11.5.5 Memory Partitions 976

- 11.6 CACHING IN WINDOWS 977

- 11.7 INPUT/OUTPUT IN WINDOWS 979
 - 11.7.1 Fundamental Concepts 980
 - 11.7.2 Input/Output API Calls 982
 - 11.7.3 Implementation of I/O 984

- 11.8 THE WINDOWS NT FILE SYSTEM 989
 - 11.8.1 Fundamental Concepts 989
 - 11.8.2 Implementation of the NT File System 990

- 11.9 WINDOWS POWER MANAGEMENT 1000

- 11.10 VIRTUALIZATION IN WINDOWS 1003
 - 11.10.1 Hyper-V 1003
 - 11.10.2 Containers 1011
 - 11.10.3 Virtualization-Based Security 1017
- 11.11 SECURITY IN WINDOWS 1018
 - 11.11.1 Fundamental Concepts 1020
 - 11.11.2 Security API Calls 1022
 - 11.11.3 Implementation of Security 1023
 - 11.11.4 Security Mitigations 1025
- 11.12 SUMMARY 1035

12 OPERATING SYSTEM DESIGN

1041

- 12.1 THE NATURE OF THE DESIGN PROBLEM 1042
 - 12.1.1 Goals 1042
 - 12.1.2 Why Is It Hard to Design an Operating System? 1043
- 12.2 INTERFACE DESIGN 1045
 - 12.2.1 Guiding Principles 1045
 - 12.2.2 Paradigms 1048
 - 12.2.3 The System-Call Interface 1051
- 12.3 IMPLEMENTATION 1053
 - 12.3.1 System Structure 1054
 - 12.3.2 Mechanism vs. Policy 1057
 - 12.3.3 Orthogonality 1058
 - 12.3.4 Naming 1059
 - 12.3.5 Binding Time 1061
 - 12.3.6 Static vs. Dynamic Structures 1062
 - 12.3.7 Top-Down vs. Bottom-Up Implementation 1063
 - 12.3.8 Synchronous vs. Asynchronous Communication 1064
 - 12.3.9 Useful Techniques 1065
- 12.4 PERFORMANCE 1070
 - 12.4.1 Why Are Operating Systems Slow? 1071
 - 12.4.2 What Should Be Optimized? 1071
 - 12.4.3 Space-Time Trade-offs 1072

- 12.4.4 Caching 1075
- 12.4.5 Hints 1076
- 12.4.6 Exploiting Locality 1077
- 12.4.7 Optimize the Common Case 1077
- 12.5 PROJECT MANAGEMENT 1078
 - 12.5.1 The Mythical Man Month 1078
 - 12.5.2 Team Structure 1079
 - 12.5.3 The Role of Experience 1081
 - 12.5.4 No Silver Bullet 1082

13 READING LIST AND BIBLIOGRAPHY 1087

- 13.1 SUGGESTIONS FOR FURTHER READING 1087
 - 13.1.1 Introduction 1088
 - 13.1.2 Processes and Threads 1088
 - 13.1.3 Memory Management 1089
 - 13.1.4 File Systems 1090
 - 13.1.5 Input/Output 1090
 - 13.1.6 Deadlocks 1091
 - 13.1.7 Virtualization and the Cloud 1092
 - 13.1.8 Multiple Processor Systems 1093
 - 13.1.9 Security 1093
 - 13.1.10 Case Study 1: UNIX, Linux, and Android 1094
 - 13.1.11 Case Study 2: Windows 1095
 - 13.1.12 Operating System Design 1096
- 13.2 ALPHABETICAL BIBLIOGRAPHY 1097

INDEX

1121

PREFACE

The fifth edition of this book differs from the fourth edition in many ways. There are large numbers of small changes everywhere to bring the material up to date as operating systems are not standing still. For example, where the previous edition focused almost exclusively on magnetic hard disks for storage, this time we give the flash-based Solid State Drives (SSDs) the attention that befits their popularity. The chapter on Windows 8.1 has been replaced entirely by a chapter on the new Windows 11. We have rewritten much of the security chapter, with more focus on topics that are directly relevant for operating systems (and exciting new attacks and defenses), while reducing the discussion of cryptography and steganography. Here is a chapter-by-chapter rundown of the changes.

- Chapter 1 has been heavily modified and updated in many places, but with the exception of dropping the description of CD-ROMs and DVDs in favor of modern storage solutions such as SSDs and persistent memory, no major sections have been added or deleted.
- In Chapter 2, we significantly expanded the discussion of event-driven servers and included an extensive example with pseudo code. We gave priority inversion its own section where we also discussed ways to deal with the problem. We reordered some of the sections to clarify the discussion. For instance, we now discuss the readers-writers problem

immediately after the producer-consumer section and moved the dining philosophers to another chapter, that of deadlocks, entirely. Besides numerous smaller updates, we also dropped some older material, such as scheduler activations and pop-up threads.

- Chapter 3 now focuses on modern 64-bit architectures and contains more precise explanations of many aspects of paging and TLBs. For instance, we describe how operating systems use paging also and how some operating systems map the kernel into the address spaces of user processes.
- Chapter 4 changed significantly. We dropped the lengthy descriptions of CD-ROMs and tapes, and instead added sections about SSD-based file systems, booting in modern UEFI-based computer systems, and secure file deletion and disk encryption.
- In Chapter 5, we have more information about SSDs and NVMe, and explain input devices using a modern USB keyboard instead of the older PS/2 one of the previous edition. In addition, we clarify the relation between interrupts, traps, exceptions, and faults.
- As mentioned, we added the dining philosophers example to Chapter 6. Other than that, the chapter is pretty much unchanged. The topic of deadlocks is fairly stable, with few new results.
- In Chapter 7, we added a section about containers to the existing (and updated) explanation of hypervisor-based virtualization. The material on VMware has also been brought up to date.
- Chapter 8 is an updated version of the previous material on multiprocessor systems. We added subsections on simultaneous multithreading and discuss new types of coprocessors, while dropping sections such as the one on the older IXP network processors and the one on the (now dead) CORBA middleware. A new section discusses scheduling for security.
- Chapter 9 has been heavily revised and reorganized, with much more focus on what is relevant for the operating system and less emphasis on crypto. We now start the chapter with a discussion of principles for secure design and their relevance for the operating system structure. We discuss exciting new hardware developments, such as the Melt-down and Spectre transient execution vulnerabilities, that have come to light since the previous edition. In addition, we describe new software vulnerabilities that are important for the operating system. Finally, we greatly expanded the description of the ways in which the operating system can be hardened, with extensive discussion of control flow integrity, fine-grained ASLR, code signing, access restrictions, and

attestation. Since there is much ongoing research in this area, new references have been added and the research section has been rewritten.

- Chapter 10 has been updated with new developments in Linux and Android. Android has evolved considerably since the previous edition, and this chapter covers the current version in detail. This section has been substantially rewritten.
- Chapter 11 has changed significantly. Where the fourth edition was on Windows 8.1, we now discuss Windows 11. It is basically a new chapter.
- Chapter 12 is a slightly revised version of the previous edition. This chapter covers the basic principles of system design, and they have not changed much in the past few years.
- Chapter 13 is a thoroughly updated list of suggested readings. In addition, the list of references has been updated, with entries to well over 100 new works published after the fourth edition of this book came out.
- In addition, the sections on research throughout the book have all been redone from scratch to reflect the latest research in operating systems. Furthermore, new problems have been added to all the chapters.

Instructor supplements (including the PowerPoint sheets) can be found at <https://www.pearsonhighered.com/cs-resources>.

A number of people have been involved in the fifth edition. The material in Chap. 7 on VMware (in Sec. 7.12) was written by Edouard Bugnion of EPFL in Lausanne, Switzerland. Ed was one of the founders of the VMware company and knows this material as well as anyone in the world. We thank him greatly for supplying it to us.

Ada Gavrilovska of Georgia Tech, who is an expert on Linux internals, updated Chap. 10 from the Fourth Edition, which she also wrote. The Android material in Chap. 10 was written by Dianne Hackborn of Google, one of the key developers of the Android system. Android is the most popular operating system on smartphones, so we are very grateful to have Dianne help us. Chapter 10 is now quite long and detailed, but UNIX, Linux, and Android fans can learn a lot from it.

We haven't neglected Windows, however. Mehmet Iyigun of Microsoft updated Chap. 11 from the previous edition of the book. This time the chapter covers Windows 11 in detail. Mehmet has a great deal of knowledge of Windows and enough vision to tell the difference between places where Microsoft got it right and where it got it wrong. He was ably assisted by Andrea Allievi, Pedro Justo, Chris Kleynhans, and Erick Smith. Windows fans are certain to enjoy this chapter.

The book is much better as a result of the work of all these expert contributors. Again, we would like to thank them for their invaluable help.

We were also fortunate to have several reviewers who read the manuscript and also suggested new end-of-chapter problems. They were Jeremiah Blanchard (University of Florida), Kate Holdener (St. Louis University), Liting Hu (Virginia Tech), Jiang-Bo Liu (Bradley University), and Mai Zheng (Iowa State University). We remain responsible for any remaining errors, of course.

We would also like to thank our editor, Tracy Johnson, for keeping the project on track and herding all the cats, albeit virtually this time. Erin Sullivan managed the review process and Carole Snyder handled production.

Finally, last but not least, Barbara, Marvin, and Matilde are still wonderful, as usual, each in a unique and special way. Aron and Nathan are great kids and Olivia and Mirte are treasures. And of course, I would like to thank Suzanne for her love and patience, not to mention all the *druiven*, *kersen*, and *sinaasappelsap*, as well as other agricultural products. (AST)

As always, I owe a massive thank you to Marieke, Duko, and Jip. Marieke for just being there and for bearing with me in the countless hours that I was working on this book, and Duko and Jip for tearing me away from it to play hoops, day and night! I am also grateful to the neighbors for putting up with our midnight basketball games. (HB)

Andrew S. Tanenbaum
Herbert Bos

ABOUT THE AUTHORS

Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California at Berkeley. He is currently a Professor Emeritus of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands. He was formerly Dean of the Advanced School for Computing and Imaging, an interuniversity graduate school doing research on advanced parallel, distributed, and imaging systems. He was also an Academy Professor of the Royal Netherlands Academy of Arts and Sciences, which has saved him from turning into a bureaucrat. He also won a prestigious European Research Council Advanced Grant.

In the past, he has done research on compilers, reliable operating systems, networking, and distributed systems. This research has led to over 200 refereed publications in journals and conferences. Prof. Tanenbaum has also authored or co-authored five books, which have been translated into over 20 languages, ranging from Basque to Thai. They are used at universities all over the world. There are 163 versions of his books.

Prof. Tanenbaum has also produced a considerable volume of software, notably MINIX, a small UNIX clone. It was the direct inspiration for Linux and the platform on which Linux was initially developed. The current version of MINIX, called MINIX 3, is now focused on being an extremely reliable and secure operating system. Prof. Tanenbaum will consider his work done when no user has any idea what an operating system crash is. MINIX 3 is an ongoing open-source project to which you are invited to contribute. Go to www.minix3.org to download a free copy of MINIX 3 and give it a try. Both x86 and ARM versions are available.

Prof. Tanenbaum's Ph.D. students have gone on to greater glory after graduating. Some have become professors; others have fulfilled leading roles in government organizations and industry. He is very proud of them. In this respect, he resembles a mother hen.

Prof. Tanenbaum is a Fellow of the ACM, a Fellow of the IEEE, and a member of the Royal Netherlands Academy of Arts and Sciences. He has also won numerous scientific prizes from ACM, IEEE, and USENIX. If you are unbearably curious about them, see his page on Wikipedia. He also has two honorary doctorates.

Herbert Bos obtained his Master's degree from Twente University and his Ph.D. from Cambridge University in the United Kingdom. Since then, he has worked extensively on dependable and efficient I/O architectures for operating systems like Linux, but also research systems based on MINIX 3. He is currently a professor at the VUsec Systems Security Research Group in the Department of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands. His main research field is systems security.

With his group, he discovered and analyzed many vulnerabilities in both hardware and software. From buggy memory chips to vulnerable CPUs, and from flaws in operating systems to novel exploitation techniques, the research has led to fixes

in most major operating systems, most popular browsers, and all modern Intel processors. He believes that offensive research is valuable because the main reason for today's security problems is that systems have become so complex that we no longer understand them. By investigating how we can make systems behave in unintended ways, we learn more about their (real) nature. Armed with this knowledge, developers can then improve their designs in the future. Indeed, while sophisticated new attacks tend to feature prominently in the media, Herbert spends most of his time on developing defensive techniques to improve the security.

His (former) students are all awesome and much cleverer than he is. With them, he has won five Pwnie Awards at the Black Hat conference in Las Vegas. Moreover, five of his students have won the ACM SIGOPS EuroSys Roger Needham Award for best European Ph.D. thesis in systems, two of them have won the ACM SIGSAC Doctoral Dissertation Award for best Ph.D. thesis in security, and two more won the William C. Carter Ph.D. Dissertation Award for their work on dependability. Herbert worries about climate change and loves the Beatles.

1

INTRODUCTION

A modern computer consists of one or more processors, some amount of main memory, hard disks or Flash drives, printers, a keyboard, a mouse, a display, network interfaces, and various other input/output devices. All in all, a complex system. If every application programmer had to understand how all these things work in detail, no code would ever get written. Furthermore, managing all these components and using them optimally is an exceedingly challenging job. For this reason, computers are equipped with a layer of software called the **operating system**, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all the resources just mentioned. Operating systems are the subject of this book.

It is important to realize that smart phones and tablets (like the Apple iPad) are just computers in a smaller package with a touch screen. They all have operating systems. In fact, Apple's iOS is fairly similar to macOS, which runs on Apple's desktop and MacBook systems. The smaller form factor and touch screen really doesn't change that much about what the operating system does. Android smartphones and tablets all run Linux as the true operating system on the bare hardware. What users perceive as "Android" is simply a layer of software running on top of Linux. Since macOS (and thus iOS) is derived from Berkeley UNIX and Linux is a clone of UNIX, by far the most popular operating system in the world is UNIX and its variants. For this reason, we will pay a lot of attention in this book to UNIX.

Most readers probably have had some experience with an operating system such as Windows, Linux, FreeBSD, or macOS, but appearances can be deceiving.