Contents

To Everyone . . . iii
To Educators . . . v
To Students . . . vii
Acknowledgments . . . viii
Final Words . . . ix
References . . . x

1 A Dialogue on the Book **1**

2 Introduction to Operating Systems **5**
2.1 Virtualizing the CPU . . . 7
2.2 Virtualizing Memory . . . 9
2.3 Concurrency . . . 11
2.4 Persistence . . . 14
2.5 Design Goals . . . 16
2.6 Some History . . . 18
2.7 Summary . . . 22
References . . . 23

I Virtualization **25**

3 A Dialogue on Virtualization **27**

4 Abstraction: The Process **29**
4.1 The Abstraction: A Process . . . 30

4.2 Process API 31
4.3 Process States 32
4.4 Data Structures 33
4.5 Summary 35
References 36

5 Interlude: Process API 37
5.1 The `fork()` System Call 37
5.2 Adding `wait()` System Call 40
5.3 Finally, the `exec()` System Call 41
5.4 Why? Motivating the API 42
5.5 Other Parts of the API 43
5.6 Summary 44
References 45

6 Mechanism: Limited Direct Execution 47
6.1 Basic Technique: Limited Direct Execution 48
6.2 Problem #1: Restricted Operations 48
6.3 Problem #2: Switching Between Processes 51
6.4 Summary 55
References 57
Homework 59

7 Scheduling: Introduction 61
7.1 Workload Assumptions 61
7.2 Scheduling Metrics 62
7.3 First In, First Out (FIFO) 63
7.4 Shortest Job First (SJF) 64
7.5 Shortest Time-to-Completion First (STCF) 66
7.6 Round Robin 68
7.7 Incorporating I/O 70
7.8 No More Oracle 72
7.9 Summary 72
References 73
Homework 74

8 Scheduling:
The Multi-Level Feedback Queue 79
8.1 MLFQ: Basic Rules 80
8.2 Attempt #1: How to Change Priority 81

8.3 Attempt #2: The Priority Boost 85
8.4 Attempt #3: Better Accounting 86
8.5 Tuning MLFQ and Other Issues 87
8.6 MLFQ: Summary 89
References 91
Homework 93

9 Scheduling: Proportional Share 99
9.1 Basic Concept: Tickets Represent Your Share 99
9.2 Ticket Mechanisms 100
9.3 Implementation 102
9.4 How To Assign Tickets? 103
9.5 Why Not Deterministic? 103
9.6 Summary 105
References 106
Homework 107

10 Multiprocessor Scheduling (INCOMPLETE) 111

11 Summary Dialogue on CPU Virtualization 113

12 The Abstraction: Address Spaces 117
12.1 Early Systems 117
12.2 Multiprogramming and Time Sharing 117
12.3 The Address Space 119
12.4 Goals 123
12.5 Summary 124
References 125

13 Interlude: Memory API 127
13.1 Types of Memory 127
13.2 The `malloc()` Call 128
13.3 The `free()` Call 130
13.4 Common Errors 131
13.5 Underlying OS Support 134
13.6 Other Calls 135
13.7 Summary 135
References 136

14 Mechanism: Address Translation 137

14.1 Assumptions 138
14.2 An Example 139
14.3 Dynamic (Hardware-based) Relocation 141
14.4 OS Issues 145
14.5 Summary 147
References 149
Homework 150

15 Segmentation 155
15.1 Segmentation: Generalized Base/Bounds 156
15.2 Which Segment Are We Referring To? 159
15.3 What About The Stack? 160
15.4 Support for Sharing 161
15.5 Fine-grained vs. Coarse-grained Segmentation 162
15.6 OS Support 163
15.7 Summary 165
References 167
Homework 169

16 Free-Space Management (INCOMPLETE) 175
16.1 Assumptions 176
16.2 Low-level Mechanisms 177
16.3 Basic Strategies 186
16.4 Other Approaches 187
16.5 Summary 189
16.6 References 190

17 Paging: Introduction 191
17.1 Where Are Page Tables Stored? 195
17.2 What's Actually In The Page Table? 196
17.3 Paging: Also Too Slow 197
References 200
Homework 201

18 Paging: Faster Translations (TLBs) 207
18.1 Who Handles the Miss? 208
18.2 TLB Contents: What's In There? 211
18.3 TLB Issue: Context Switches 212
18.4 Issue: Replacement Policy 214
18.5 Real Code 215

18.6 An Example 216
18.7 Summary 217
References 219
Homework 221

19 Paging: Smaller Tables 225
19.1 Simple Solution: Bigger Pages 225
19.2 Hybrid Approach: Paging and Segments 226
19.3 Multi-level Page Tables 230
19.4 Inverted Page Tables 239
19.5 Swapping the Page Tables to Disk 240
19.6 Paging: Summary 240
References 241
Homework 242

20 Beyond Physical Memory: Mechanisms 245
20.1 Swap Space 246
20.2 The Present Bit 247
20.3 The Page Fault 248
20.4 What If Memory Is Full? 249
20.5 Page Fault Control Flow 250
20.6 When Replacements Really Occur 251
20.7 Summary 252
References 254

21 Beyond Physical Memory: Policies 255
21.1 Cache Management 256
21.2 The Optimal Replacement Policy 257
21.3 A Simple Policy: FIFO 259
21.4 Another Simple Policy: Random 260
21.5 Using History: LRU 262
21.6 Workload Examples 264
21.7 Implementing Historical Algorithms 266
21.8 Approximating LRU 267
21.9 Considering Dirty Pages 269
21.10 Other VM Policies 269
21.11 Thrashing 270
21.12 Summary 271
References 272
Homework 274

22 Case Study: The VAX/VMS Virtual Memory System **279**
22.1 Background 279
22.2 Memory Management Hardware 280
22.3 A Real Address Space 283
22.4 Page Replacement 284
22.5 Other Neat VM Tricks 286
22.6 Summary 287
References 288

23 Summary Dialogue on Memory Virtualization **289**

II Concurrency **293**

24 A Dialogue on Concurrency **295**

25 Concurrency: An Introduction **297**
25.1 An Example: Thread Creation 299
25.2 Why It Gets Worse: Shared Data 301
25.3 The Heart of the Problem: Uncontrolled Scheduling . 304
25.4 The Wish For Atomicity 307
25.5 One More Problem: Waiting For Another 309
25.6 Summary: Why in OS Class? 309
References 311

26 Interlude: Thread API **313**
26.1 Thread Creation 313
26.2 Thread Completion 315
26.3 Locks 318
26.4 Condition Variables 320
26.5 Compiling and Running 322
26.6 Summary 323
References 325

27 Locks **327**
27.1 Locks: The Basic Idea 327
27.2 Pthread Locks 328
27.3 Building A Lock 329
27.4 Evaluating Locks 330
27.5 Controlling Interrupts 330

27.6 Test And Set (Atomic Exchange) 333
27.7 Building A Working Spin Lock 335
27.8 Evaluating Spin Locks 336
27.9 Compare-And-Swap 337
27.10 Load-Linked and Store-Conditional 339
27.11 Fetch-And-Add 340
27.12 Summary: So Much Spinning 342
27.13 A Simple Approach: Just Yield, Baby 343
27.14 Using Queues: Sleeping Instead Of Spinning 344
27.15 Different OS, Different Support 347
27.16 Two-Phase Locks 348
27.17 Summary 349
References 350

28 Using Locks (INCOMPLETE) 353
28.1 Concurrent Counter 353
28.2 Concurrent Linked List 354
28.3 Concurrent Hash Table 355

29 Condition Variables 357
29.1 Definition and Routines 359
29.2 The Producer/Consumer (Bound Buffer) Problem . . . 362
29.3 Covering Conditions 371
29.4 Summary 372
References 373

30 Semaphores 375
30.1 Semaphores: A Definition 375
30.2 Binary Semaphores (Locks) 377
30.3 Semaphores As Condition Variables 378
30.4 The Producer/Consumer (Bounded-Buffer) Problem . 380
30.5 Reader-Writer Locks 384
30.6 The Dining Philosophers 386
30.7 How To Implement Semaphores 389
30.8 Summary 390
References 392

31 Deadlock 395
31.1 Why Do Deadlocks Occur? 396
31.2 Conditions for Deadlock 397

31.3 Prevention 398
31.4 Avoidance via Scheduling 402
31.5 Detect and Recover 403
31.6 Summary 404
References 405

32 Real-World Concurrency Bugs (INCOMPLETE) 407
References 409

33 Summary Dialogue on Concurrency 411

III Persistence 413

34 A Dialogue on Persistence 415

35 I/O Devices 417
35.1 System Architecture 417
35.2 A Canonical Device 418
35.3 The Canonical Protocol 419
35.4 Lowering CPU Overhead with Interrupts 420
35.5 More Efficient Data Movement with DMA 422
35.6 Methods of Device Interaction 423
35.7 Fitting into the OS: The Device Driver 424
35.8 Case Study: A Simple IDE Disk Driver 426
35.9 Historical Notes 429
35.10 Summary 430
References 431

36 Hard Disk Drives 433
36.1 The Interface 433
36.2 Basic Geometry 434
36.3 A Simple Disk Drive 435
36.4 I/O Time: Doing The Math 439
36.5 Disk Scheduling 443
36.6 Summary 447
References 448
Homework 449

37 Redundant Arrays of Inexpensive Disks (RAIDs) 455

37.1 Interface and RAID Internals 456
37.2 Fault Model 457
37.3 How to Evaluate a RAID 458
37.4 RAID Level 0: Striping 458
37.5 RAID Level 1: Mirroring 463
37.6 RAID Level 4: Saving Space with Parity 466
37.7 RAID Level 5: Rotating Parity 471
37.8 RAID Comparison: A Summary 472
37.9 Other Interesting RAID Issues 473
37.10 Summary 474
References 475
Homework 477

38 The Abstraction: Files and Directories 483
38.1 Files and Directories 484
38.2 The File System Interface 486
38.3 Creating Files 486
38.4 Reading and Writing Files 487
38.5 Reading And Writing, But Not Sequentially 490
38.6 Getting Information About Files 490
38.7 Removing Files 492
38.8 Making Directories 493
38.9 Reading Directories 494
38.10 Deleting Directories 495
38.11 Hard Links 495
38.12 Symbolic Links 497
38.13 Making and Mounting a File System 499
38.14 Summary 501
References 502
Homework 503

39 File System Implementation 505
39.1 The Way To Think 506
39.2 Overall Organization 506
39.3 File Organization: The Inode 509
39.4 Directory Organization 514
39.5 Free Space Management 515
39.6 Access Paths: Reading and Writing 515
39.7 Caching and Buffering 520
39.8 Summary 521

References . . . 522
Homework . . . 524

40 Locality and The Fast File System 531
40.1 The Problem: Poor Performance . . . 531
40.2 FFS: Disk Awareness Is The Solution . . . 533
40.3 Organizing Structure: The Cylinder Group . . . 533
40.4 Policies: How To Allocate Files and Directories . . . 535
40.5 The Large-File Exception . . . 535
40.6 A Few Other Things About FFS . . . 537
40.7 Summary . . . 540
References . . . 541

41 Crash Consistency: FSCK and Journaling 543
41.1 A Detailed Example . . . 544
41.2 Solution #1: The File System Checker . . . 548
41.3 Solution #2: Journaling (or Write-Ahead Logging) . . . 550
41.4 Solution #3: Other Approaches . . . 560
41.5 Summary . . . 561
References . . . 562

42 Log-structured File Systems 565
42.1 Writing To The Log: Some Details . . . 566
42.2 How Can We Find Those Pesky Inodes? . . . 567
42.3 Solution Through Indirection: The Inode Map . . . 568
42.4 Even More Problems: Where To Put The Inode Map? . 569
42.5 Reading A File From Disk: A Recap . . . 570
42.6 A New Problem: Garbage Collection . . . 571
42.7 How Can We Determine Which Blocks Are Live? . . . 571
42.8 A Policy Question: Which Blocks To Clean, And When? 572
42.9 Crash Recovery . . . 573
42.10 Summary . . . 573
References . . . 575

43 Modern File Systems (INCOMPLETE) 577
43.1 WAFL and Checksums . . . 577
43.2 XFS and B-Trees . . . 577
43.3 ZFS and RAID . . . 577
References . . . 578

44 Summary Dialogue on Persistence **579**

45 A Dialogue on Distribution **581**

46 Distributed Systems **583**
46.1 Communication Basics 584
46.2 Unreliable Communication Layers 585
46.3 Reliable Communication Layers 588
46.4 Communication Abstractions 593
46.5 Remote Procedure Call (RPC) 594
46.6 Summary 600
References 601

47 Sun's Network File System (NFS) **603**
47.1 A Basic Distributed File System 603
47.2 On To NFS 605
47.3 Focus: Simple and Fast Server Crash Recovery 605
47.4 Key To Fast Crash Recovery: Statelessness 606
47.5 The NFSv2 Protocol 608
47.6 From Protocol to Distributed File System 610
47.7 Handling Server Failure with Idempotent Operations . 612
47.8 Improving Performance: Client-side Caching 614
47.9 The Cache Consistency Problem 615
47.10 Assessing NFS Cache Consistency 617
47.11 Implications on Server-Side Write Buffering 617
47.12 Summary 619
References 621

48 The Andrew File System (AFS) **623**
48.1 AFS Version 1 623
48.2 Problems with Version 1 625
48.3 Improving the Protocol 626
48.4 AFS Version 2 626
48.5 Cache Consistency 627
48.6 Crash Recovery 628
48.7 Scale of AFSv2 629
48.8 Other Improvements: Namespaces, Security, Etc. . . . 629
48.9 Summary 629
References 630

V Appendices **631**

49 A Dialogue on Virtual Machine Monitors **633**

50 Virtual Machine Monitors **635**
50.1 Introduction 635
50.2 Motivation: Why VMMs? 636
50.3 Virtualizing the CPU 637
50.4 Virtualizing Memory 641
50.5 The Information Gap 646
50.6 Summary 647
References 649

51 VMM Memory Management (INCOMPLETE) **651**
51.1 Introduction 651
References 651

General Index **653**

Advice, Asides, Design Tips, Data Structures, and Hardware Support **665**

Cruces **669**