Contents

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

1 A Dialogue on the Book **1**

2 Introduction to Operating Systems **3**
2.1 Virtualizing the CPU . . . 5
2.2 Virtualizing Memory . . . 7
2.3 Concurrency . . . 8
2.4 Persistence . . . 11
2.5 Design Goals . . . 13
2.6 Some History . . . 14
2.7 Summary . . . 18
References . . . 19

I Virtualization **21**

3 A Dialogue on Virtualization **23**

4 The Abstraction: The Process **25**
4.1 The Abstraction: A Process . . . 26
4.2 Process API . . . 27
4.3 Process Creation: A Little More Detail . . . 28
4.4 Process States . . . 29
4.5 Data Structures . . . 31
4.6 Summary . . . 33
References . . . 34
Homework . . . 35

5 Interlude: Process API **37**
5.1 The `fork()` System Call . . . 37
5.2 The `wait()` System Call . . . 39
5.3 Finally, The `exec()` System Call . . . 40
5.4 Why? Motivating The API . . . 41
5.5 Other Parts Of The API . . . 44
5.6 Summary . . . 44
References . . . 45
Homework (Code) . . . 46

6 Mechanism: Limited Direct Execution **49**
6.1 Basic Technique: Limited Direct Execution . . . 49
6.2 Problem #1: Restricted Operations . . . 50
6.3 Problem #2: Switching Between Processes . . . 54
6.4 Worried About Concurrency? . . . 58
6.5 Summary . . . 59
References . . . 61
Homework (Measurement) . . . 62

7 Scheduling: Introduction **63**
7.1 Workload Assumptions . . . 63
7.2 Scheduling Metrics . . . 64
7.3 First In, First Out (FIFO) . . . 64
7.4 Shortest Job First (SJF) . . . 66
7.5 Shortest Time-to-Completion First (STCF) . . . 67
7.6 A New Metric: Response Time . . . 68
7.7 Round Robin . . . 69
7.8 Incorporating I/O . . . 71
7.9 No More Oracle . . . 72
7.10 Summary . . . 72
References . . . 73
Homework . . . 74

8 Scheduling:
The Multi-Level Feedback Queue **75**
8.1 MLFQ: Basic Rules . . . 76
8.2 Attempt #1: How To Change Priority . . . 77
8.3 Attempt #2: The Priority Boost . . . 80
8.4 Attempt #3: Better Accounting . . . 81
8.5 Tuning MLFQ And Other Issues . . . 82
8.6 MLFQ: Summary . . . 83
References . . . 85
Homework . . . 86

9 Scheduling: Proportional Share **87**
9.1 Basic Concept: Tickets Represent Your Share . . . 87
9.2 Ticket Mechanisms . . . 89

9.3 Implementation . . . 90
9.4 An Example . . . 91
9.5 How To Assign Tickets? . . . 92
9.6 Why Not Deterministic? . . . 92
9.7 Summary . . . 93
References . . . 95
Homework . . . 96

10 Multiprocessor Scheduling (Advanced) 97
10.1 Background: Multiprocessor Architecture . . . 98
10.2 Don't Forget Synchronization . . . 100
10.3 One Final Issue: Cache Affinity . . . 101
10.4 Single-Queue Scheduling . . . 101
10.5 Multi-Queue Scheduling . . . 103
10.6 Linux Multiprocessor Schedulers . . . 106
10.7 Summary . . . 106
References . . . 107

11 Summary Dialogue on CPU Virtualization 109

12 A Dialogue on Memory Virtualization 111

13 The Abstraction: Address Spaces 113
13.1 Early Systems . . . 113
13.2 Multiprogramming and Time Sharing . . . 114
13.3 The Address Space . . . 115
13.4 Goals . . . 117
13.5 Summary . . . 119
References . . . 120

14 Interlude: Memory API 123
14.1 Types of Memory . . . 123
14.2 The `malloc()` Call . . . 124
14.3 The `free()` Call . . . 126
14.4 Common Errors . . . 126
14.5 Underlying OS Support . . . 129
14.6 Other Calls . . . 130
14.7 Summary . . . 130
References . . . 131
Homework (Code) . . . 132

15 Mechanism: Address Translation 135
15.1 Assumptions . . . 136
15.2 An Example . . . 136
15.3 Dynamic (Hardware-based) Relocation . . . 139
15.4 Hardware Support: A Summary . . . 142
15.5 Operating System Issues . . . 143

© 2014, ARPACI-DUSSEAU

15.6 Summary . . . 146
References . . . 147
Homework . . . 148

16 Segmentation 149
16.1 Segmentation: Generalized Base/Bounds . . . 149
16.2 Which Segment Are We Referring To? . . . 152
16.3 What About The Stack? . . . 153
16.4 Support for Sharing . . . 154
16.5 Fine-grained vs. Coarse-grained Segmentation . . . 155
16.6 OS Support . . . 155
16.7 Summary . . . 157
References . . . 158
Homework . . . 160

17 Free-Space Management 161
17.1 Assumptions . . . 162
17.2 Low-level Mechanisms . . . 163
17.3 Basic Strategies . . . 171
17.4 Other Approaches . . . 173
17.5 Summary . . . 175
References . . . 176
Homework . . . 177

18 Paging: Introduction 179
18.1 A Simple Example And Overview . . . 179
18.2 Where Are Page Tables Stored? . . . 183
18.3 What's Actually In The Page Table? . . . 184
18.4 Paging: Also Too Slow . . . 185
18.5 A Memory Trace . . . 186
18.6 Summary . . . 189
References . . . 190
Homework . . . 191

19 Paging: Faster Translations (TLBs) 193
19.1 TLB Basic Algorithm . . . 193
19.2 Example: Accessing An Array . . . 195
19.3 Who Handles The TLB Miss? . . . 197
19.4 TLB Contents: What's In There? . . . 199
19.5 TLB Issue: Context Switches . . . 200
19.6 Issue: Replacement Policy . . . 202
19.7 A Real TLB Entry . . . 203
19.8 Summary . . . 204
References . . . 205
Homework (Measurement) . . . 207

20 Paging: Smaller Tables 211

20.1 Simple Solution: Bigger Pages . . . 211
20.2 Hybrid Approach: Paging and Segments . . . 212
20.3 Multi-level Page Tables . . . 215
20.4 Inverted Page Tables . . . 222
20.5 Swapping the Page Tables to Disk . . . 223
20.6 Summary . . . 223
References . . . 224
Homework . . . 225

21 Beyond Physical Memory: Mechanisms 227
21.1 Swap Space . . . 228
21.2 The Present Bit . . . 229
21.3 The Page Fault . . . 230
21.4 What If Memory Is Full? . . . 231
21.5 Page Fault Control Flow . . . 232
21.6 When Replacements Really Occur . . . 233
21.7 Summary . . . 234
References . . . 235

22 Beyond Physical Memory: Policies 237
22.1 Cache Management . . . 237
22.2 The Optimal Replacement Policy . . . 238
22.3 A Simple Policy: FIFO . . . 240
22.4 Another Simple Policy: Random . . . 242
22.5 Using History: LRU . . . 243
22.6 Workload Examples . . . 244
22.7 Implementing Historical Algorithms . . . 247
22.8 Approximating LRU . . . 248
22.9 Considering Dirty Pages . . . 249
22.10 Other VM Policies . . . 250
22.11 Thrashing . . . 250
22.12 Summary . . . 251
References . . . 252
Homework . . . 254

23 The VAX/VMS Virtual Memory System 255
23.1 Background . . . 255
23.2 Memory Management Hardware . . . 256
23.3 A Real Address Space . . . 257
23.4 Page Replacement . . . 259
23.5 Other Neat VM Tricks . . . 260
23.6 Summary . . . 262
References . . . 263

24 Summary Dialogue on Memory Virtualization 265

© 2014, ARPACI-DUSSEAU

II Concurrency **269**

25 A Dialogue on Concurrency **271**

26 Concurrency: An Introduction **273**
26.1 An Example: Thread Creation . . . 274
26.2 Why It Gets Worse: Shared Data . . . 277
26.3 The Heart Of The Problem: Uncontrolled Scheduling . . . 279
26.4 The Wish For Atomicity . . . 281
26.5 One More Problem: Waiting For Another . . . 283
26.6 Summary: Why in OS Class? . . . 283
References . . . 285
Homework . . . 286

27 Interlude: Thread API **289**
27.1 Thread Creation . . . 289
27.2 Thread Completion . . . 290
27.3 Locks . . . 293
27.4 Condition Variables . . . 295
27.5 Compiling and Running . . . 297
27.6 Summary . . . 297
References . . . 299

28 Locks **301**
28.1 Locks: The Basic Idea . . . 301
28.2 Pthread Locks . . . 302
28.3 Building A Lock . . . 303
28.4 Evaluating Locks . . . 303
28.5 Controlling Interrupts . . . 304
28.6 Test And Set (Atomic Exchange) . . . 306
28.7 Building A Working Spin Lock . . . 307
28.8 Evaluating Spin Locks . . . 309
28.9 Compare-And-Swap . . . 309
28.10 Load-Linked and Store-Conditional . . . 311
28.11 Fetch-And-Add . . . 312
28.12 Too Much Spinning: What Now? . . . 313
28.13 A Simple Approach: Just Yield, Baby . . . 314
28.14 Using Queues: Sleeping Instead Of Spinning . . . 315
28.15 Different OS, Different Support . . . 317
28.16 Two-Phase Locks . . . 318
28.17 Summary . . . 319
References . . . 320
Homework . . . 322

29 Lock-based Concurrent Data Structures **325**
29.1 Concurrent Counters . . . 325
29.2 Concurrent Linked Lists . . . 330

29.3 Concurrent Queues . . . 333
29.4 Concurrent Hash Table . . . 334
29.5 Summary . . . 336
References . . . 337

30 Condition Variables 339
30.1 Definition and Routines . . . 340
30.2 The Producer/Consumer (Bounded Buffer) Problem . . . 343
30.3 Covering Conditions . . . 351
30.4 Summary . . . 352
References . . . 353

31 Semaphores 355
31.1 Semaphores: A Definition . . . 355
31.2 Binary Semaphores (Locks) . . . 357
31.3 Semaphores As Condition Variables . . . 358
31.4 The Producer/Consumer (Bounded Buffer) Problem . . . 360
31.5 Reader-Writer Locks . . . 364
31.6 The Dining Philosophers . . . 366
31.7 How To Implement Semaphores . . . 369
31.8 Summary . . . 370
References . . . 371

32 Common Concurrency Problems 373
32.1 What Types Of Bugs Exist? . . . 373
32.2 Non-Deadlock Bugs . . . 374
32.3 Deadlock Bugs . . . 377
32.4 Summary . . . 385
References . . . 386

33 Event-based Concurrency (Advanced) 389
33.1 The Basic Idea: An Event Loop . . . 389
33.2 An Important API: `select()` (or `poll()`) . . . 390
33.3 Using `select()` . . . 391
33.4 Why Simpler? No Locks Needed . . . 392
33.5 A Problem: Blocking System Calls . . . 393
33.6 A Solution: Asynchronous I/O . . . 393
33.7 Another Problem: State Management . . . 396
33.8 What Is Still Difficult With Events . . . 397
33.9 Summary . . . 397
References . . . 398

34 Summary Dialogue on Concurrency 399

© 2014, ARPACI-DUSSEAU

III Persistence **401**

35 A Dialogue on Persistence **403**

36 I/O Devices **405**
36.1 System Architecture . . . 405
36.2 A Canonical Device . . . 406
36.3 The Canonical Protocol . . . 407
36.4 Lowering CPU Overhead With Interrupts . . . 408
36.5 More Efficient Data Movement With DMA . . . 409
36.6 Methods Of Device Interaction . . . 410
36.7 Fitting Into The OS: The Device Driver . . . 411
36.8 Case Study: A Simple IDE Disk Driver . . . 412
36.9 Historical Notes . . . 415
36.10 Summary . . . 415
References . . . 416

37 Hard Disk Drives **419**
37.1 The Interface . . . 419
37.2 Basic Geometry . . . 420
37.3 A Simple Disk Drive . . . 421
37.4 I/O Time: Doing The Math . . . 424
37.5 Disk Scheduling . . . 428
37.6 Summary . . . 432
References . . . 433
Homework . . . 434

38 Redundant Arrays of Inexpensive Disks (RAIDs) **437**
38.1 Interface And RAID Internals . . . 438
38.2 Fault Model . . . 439
38.3 How To Evaluate A RAID . . . 439
38.4 RAID Level 0: Striping . . . 440
38.5 RAID Level 1: Mirroring . . . 443
38.6 RAID Level 4: Saving Space With Parity . . . 446
38.7 RAID Level 5: Rotating Parity . . . 450
38.8 RAID Comparison: A Summary . . . 451
38.9 Other Interesting RAID Issues . . . 452
38.10 Summary . . . 452
References . . . 453
Homework . . . 455

39 Interlude: File and Directories **457**
39.1 Files and Directories . . . 457
39.2 The File System Interface . . . 459
39.3 Creating Files . . . 459
39.4 Reading and Writing Files . . . 460
39.5 Reading And Writing, But Not Sequentially . . . 462

39.6 Writing Immediately with `fsync()` . . . 463
39.7 Renaming Files . . . 464
39.8 Getting Information About Files . . . 465
39.9 Removing Files . . . 466
39.10 Making Directories . . . 466
39.11 Reading Directories . . . 467
39.12 Deleting Directories . . . 468
39.13 Hard Links . . . 468
39.14 Symbolic Links . . . 470
39.15 Making and Mounting a File System . . . 472
39.16 Summary . . . 473
References . . . 474
Homework . . . 475

40 File System Implementation 477
40.1 The Way To Think . . . 477
40.2 Overall Organization . . . 478
40.3 File Organization: The Inode . . . 480
40.4 Directory Organization . . . 485
40.5 Free Space Management . . . 485
40.6 Access Paths: Reading and Writing . . . 486
40.7 Caching and Buffering . . . 490
40.8 Summary . . . 492
References . . . 493
Homework . . . 494

41 Locality and The Fast File System 495
41.1 The Problem: Poor Performance . . . 495
41.2 FFS: Disk Awareness Is The Solution . . . 497
41.3 Organizing Structure: The Cylinder Group . . . 497
41.4 Policies: How To Allocate Files and Directories . . . 498
41.5 Measuring File Locality . . . 499
41.6 The Large-File Exception . . . 500
41.7 A Few Other Things About FFS . . . 502
41.8 Summary . . . 504
References . . . 505

42 Crash Consistency: FSCK and Journaling 507
42.1 A Detailed Example . . . 508
42.2 Solution #1: The File System Checker . . . 511
42.3 Solution #2: Journaling (or Write-Ahead Logging) . . . 513
42.4 Solution #3: Other Approaches . . . 523
42.5 Summary . . . 524
References . . . 525

43 Log-structured File Systems 527
43.1 Writing To Disk Sequentially . . . 528

© 2014, ARPACI-DUSSEAU

43.2 Writing Sequentially And Effectively . . . 529
43.3 How Much To Buffer? . . . 530
43.4 Problem: Finding Inodes . . . 531
43.5 Solution Through Indirection: The Inode Map . . . 531
43.6 The Checkpoint Region . . . 532
43.7 Reading A File From Disk: A Recap . . . 533
43.8 What About Directories? . . . 533
43.9 A New Problem: Garbage Collection . . . 534
43.10 Determining Block Liveness . . . 536
43.11 A Policy Question: Which Blocks To Clean, And When? . . . 537
43.12 Crash Recovery And The Log . . . 537
43.13 Summary . . . 538
References . . . 540

44 Data Integrity and Protection 543
44.1 Disk Failure Modes . . . 543
44.2 Handling Latent Sector Errors . . . 545
44.3 Detecting Corruption: The Checksum . . . 546
44.4 Using Checksums . . . 549
44.5 A New Problem: Misdirected Writes . . . 550
44.6 One Last Problem: Lost Writes . . . 551
44.7 Scrubbing . . . 551
44.8 Overheads Of Checksumming . . . 552
44.9 Summary . . . 552
References . . . 553

45 Summary Dialogue on Persistence 555

46 A Dialogue on Distribution 557

47 Distributed Systems 559
47.1 Communication Basics . . . 560
47.2 Unreliable Communication Layers . . . 561
47.3 Reliable Communication Layers . . . 563
47.4 Communication Abstractions . . . 565
47.5 Remote Procedure Call (RPC) . . . 567
47.6 Summary . . . 572
References . . . 573

48 Sun's Network File System (NFS) 575
48.1 A Basic Distributed File System . . . 576
48.2 On To NFS . . . 577
48.3 Focus: Simple and Fast Server Crash Recovery . . . 577
48.4 Key To Fast Crash Recovery: Statelessness . . . 578
48.5 The NFSv2 Protocol . . . 579
48.6 From Protocol to Distributed File System . . . 581
48.7 Handling Server Failure with Idempotent Operations . . . 583

48.8 Improving Performance: Client-side Caching 585
48.9 The Cache Consistency Problem 585
48.10 Assessing NFS Cache Consistency 587
48.11 Implications on Server-Side Write Buffering 587
48.12 Summary 589
References 590

49 The Andrew File System (AFS) 591
49.1 AFS Version 1 591
49.2 Problems with Version 1 592
49.3 Improving the Protocol 594
49.4 AFS Version 2 594
49.5 Cache Consistency 596
49.6 Crash Recovery 598
49.7 Scale And Performance Of AFSv2 598
49.8 AFS: Other Improvements 600
49.9 Summary 601
References 603
Homework 604

50 Summary Dialogue on Distribution 605

General Index 607

Asides 617

Tips 619

Cruces 621

© 2014, ARPACI-DUSSEAU