
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	10
2.4 Persistence	12
2.5 Design Goals	15
2.6 Some History	16
2.7 Summary	21
References	23
I Virtualization	25
3 A Dialogue on Virtualization	27
4 The 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 <code>fork()</code> System Call	37
5.2	Adding <code>wait()</code> System Call	40
5.3	Finally, the <code>exec()</code> 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 (Measurement)	58
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	75
8.1	MLFQ: Basic Rules	76
8.2	Attempt #1: How to Change Priority	77

8.3	Attempt #2: The Priority Boost	81
8.4	Attempt #3: Better Accounting	82
8.5	Tuning MLFQ And Other Issues	83
8.6	MLFQ: Summary	85
	References	87
	Homework	89
9	Scheduling: Proportional Share	91
9.1	Basic Concept: Tickets Represent Your Share	91
9.2	Ticket Mechanisms	92
9.3	Implementation	94
9.4	An Example	96
9.5	How To Assign Tickets?	97
9.6	Why Not Deterministic?	97
9.7	Summary	99
	References	100
	Homework	101
10	Multiprocessor Scheduling (Advanced)	103
10.1	Background: Multiprocessor Architecture	104
10.2	Don't Forget Synchronization	107
10.3	One Final Issue: Cache Affinity	108
10.4	Single-Queue Scheduling	108
10.5	Multi-Queue Scheduling	110
10.6	Linux Multiprocessor Schedulers	114
10.7	Summary	114
	References	115
11	Summary Dialogue on CPU Virtualization	117
12	The Abstraction: Address Spaces	121
12.1	Early Systems	121
12.2	Multiprogramming and Time Sharing	121
12.3	The Address Space	123
12.4	Goals	127
12.5	Summary	128
	References	129
13	Interlude: Memory API	131
13.1	Types of Memory	131

13.2	The <code>malloc()</code> Call	132
13.3	The <code>free()</code> Call	134
13.4	Common Errors	135
13.5	Underlying OS Support	138
13.6	Other Calls	139
13.7	Summary	139
	References	140
14	Mechanism: Address Translation	141
14.1	Assumptions	142
14.2	An Example	143
14.3	Dynamic (Hardware-based) Relocation	146
14.4	OS Issues	150
14.5	Summary	151
	References	153
	Homework	154
15	Segmentation	155
15.1	Segmentation: Generalized Base/Bounds	157
15.2	Which Segment Are We Referring To?	159
15.3	What About The Stack?	161
15.4	Support for Sharing	162
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	171
16.1	Assumptions	172
16.2	Low-level Mechanisms	173
16.3	Basic Strategies	182
16.4	Other Approaches	184
16.5	Summary	187
	References	189
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	198

17.4 Summary	200
References	201
Homework	202
18 Paging: Faster Translations (TLBs)	205
18.1 Who Handles The TLB Miss?	206
18.2 TLB Contents: What's In There?	209
18.3 TLB Issue: Context Switches	210
18.4 Issue: Replacement Policy	212
18.5 Real Code	213
18.6 An Example	214
18.7 Summary	215
References	217
Homework (Measurement)	219
19 Paging: Smaller Tables	223
19.1 Simple Solution: Bigger Pages	224
19.2 Hybrid Approach: Paging and Segments	225
19.3 Multi-level Page Tables	228
19.4 Inverted Page Tables	238
19.5 Swapping the Page Tables to Disk	238
19.6 Summary	238
References	240
Homework	241
20 Beyond Physical Memory: Mechanisms	243
20.1 Swap Space	244
20.2 The Present Bit	245
20.3 The Page Fault	246
20.4 What If Memory Is Full?	247
20.5 Page Fault Control Flow	248
20.6 When Replacements Really Occur	249
20.7 Summary	250
References	252
21 Beyond Physical Memory: Policies	253
21.1 Cache Management	254
21.2 The Optimal Replacement Policy	255
21.3 A Simple Policy: FIFO	257
21.4 Another Simple Policy: Random	258

21.5	Using History: LRU	260
21.6	Workload Examples	262
21.7	Implementing Historical Algorithms	265
21.8	Approximating LRU	266
21.9	Considering Dirty Pages	268
21.10	Other VM Policies	268
21.11	Thrashing	269
21.12	Summary	270
	References	271
	Homework	273
22	Case Study: The VAX/VMS Virtual Memory System	275
22.1	Background	275
22.2	Memory Management Hardware	276
22.3	A Real Address Space	278
22.4	Page Replacement	280
22.5	Other Neat VM Tricks	282
22.6	Summary	284
	References	285
23	Summary Dialogue on Memory Virtualization	287
II	Concurrency	291
24	A Dialogue on Concurrency	293
25	Concurrency: An Introduction	295
25.1	An Example: Thread Creation	297
25.2	Why It Gets Worse: Shared Data	300
25.3	The Heart of the Problem: Uncontrolled Scheduling	302
25.4	The Wish For Atomicity	305
25.5	One More Problem: Waiting For Another	307
25.6	Summary: Why in OS Class?	307
	References	309
26	Interlude: Thread API	311
26.1	Thread Creation	311
26.2	Thread Completion	313
26.3	Locks	316

26.4	Condition Variables	318
26.5	Compiling and Running	321
26.6	Summary	321
	References	323
27	Locks	325
27.1	Locks: The Basic Idea	325
27.2	Pthread Locks	326
27.3	Building A Lock	327
27.4	Evaluating Locks	328
27.5	Controlling Interrupts	328
27.6	Test And Set (Atomic Exchange)	331
27.7	Building A Working Spin Lock	333
27.8	Evaluating Spin Locks	334
27.9	Compare-And-Swap	335
27.10	Load-Linked and Store-Conditional	337
27.11	Fetch-And-Add	338
27.12	Summary: So Much Spinning	340
27.13	A Simple Approach: Just Yield, Baby	341
27.14	Using Queues: Sleeping Instead Of Spinning	342
27.15	Different OS, Different Support	345
27.16	Two-Phase Locks	346
27.17	Summary	347
	References	348
28	Lock-based Concurrent Data Structures	351
28.1	Concurrent Counters	351
28.2	Concurrent Linked Lists	357
28.3	Concurrent Queues	361
28.4	Concurrent Hash Table	363
28.5	Summary	364
	References	366
29	Condition Variables	369
29.1	Definition and Routines	371
29.2	The Producer/Consumer (Bound Buffer) Problem	374
29.3	Covering Conditions	383
29.4	Summary	384
	References	385

30	Semaphores	387
30.1	Semaphores: A Definition	388
30.2	Binary Semaphores (Locks)	389
30.3	Semaphores As Condition Variables	390
30.4	The Producer/Consumer (Bounded-Buffer) Problem	392
30.5	Reader-Writer Locks	397
30.6	The Dining Philosophers	399
30.7	How To Implement Semaphores	402
30.8	Summary	403
	References	404
31	Common Concurrency Problems	407
31.1	What Types Of Bugs Exist?	407
31.2	Non-Deadlock Bugs	408
31.3	Deadlock Bugs	412
31.4	Summary	421
	References	422
32	Event-based Concurrency (Advanced)	423
32.1	The Basic Idea: An Event Loop	424
32.2	An Important API: <code>select()</code> (or <code>poll()</code>)	424
32.3	Using <code>select()</code>	426
32.4	Why Simpler? No Locks Needed	427
32.5	A Problem: Blocking System Calls	427
32.6	A Solution: Asynchronous I/O	428
32.7	Another Problem: State Management	430
32.8	What Is Still Difficult With Events	431
32.9	Summary	432
	References	433
33	Summary Dialogue on Concurrency	435
III	Persistence	437
34	A Dialogue on Persistence	439
35	I/O Devices	441
35.1	System Architecture	441
35.2	A Canonical Device	442

35.3	The Canonical Protocol	443
35.4	Lowering CPU Overhead With Interrupts	444
35.5	More Efficient Data Movement With DMA	446
35.6	Methods Of Device Interaction	447
35.7	Fitting Into The OS: The Device Driver	448
35.8	Case Study: A Simple IDE Disk Driver	450
35.9	Historical Notes	453
35.10	Summary	454
	References	455
36	Hard Disk Drives	457
36.1	The Interface	457
36.2	Basic Geometry	458
36.3	A Simple Disk Drive	459
36.4	I/O Time: Doing The Math	463
36.5	Disk Scheduling	467
36.6	Summary	472
	References	473
	Homework	474
37	Redundant Arrays of Inexpensive Disks (RAIDs)	477
37.1	Interface And RAID Internals	479
37.2	Fault Model	479
37.3	How To Evaluate A RAID	480
37.4	RAID Level 0: Striping	481
37.5	RAID Level 1: Mirroring	485
37.6	RAID Level 4: Saving Space With Parity	488
37.7	RAID Level 5: Rotating Parity	493
37.8	RAID Comparison: A Summary	494
37.9	Other Interesting RAID Issues	495
37.10	Summary	496
	References	497
	Homework	499
38	The Abstraction: Files and Directories	501
38.1	Files and Directories	502
38.2	The File System Interface	504
38.3	Creating Files	504
38.4	Reading and Writing Files	505
38.5	Reading And Writing, But Not Sequentially	508

38.6	Getting Information About Files	509
38.7	Removing Files	510
38.8	Making Directories	511
38.9	Reading Directories	512
38.10	Deleting Directories	513
38.11	Hard Links	513
38.12	Symbolic Links	515
38.13	Making and Mounting a File System	517
38.14	Summary	519
	References	520
	Homework	521
39	File System Implementation	523
39.1	The Way To Think	524
39.2	Overall Organization	524
39.3	File Organization: The Inode	527
39.4	Directory Organization	532
39.5	Free Space Management	533
39.6	Access Paths: Reading and Writing	534
39.7	Caching and Buffering	538
39.8	Summary	540
	References	541
	Homework	543
40	Locality and The Fast File System	545
40.1	The Problem: Poor Performance	545
40.2	FFS: Disk Awareness Is The Solution	547
40.3	Organizing Structure: The Cylinder Group	547
40.4	Policies: How To Allocate Files and Directories	548
40.5	Measuring File Locality	550
40.6	The Large-File Exception	551
40.7	A Few Other Things About FFS	553
40.8	Summary	555
	References	556
41	Crash Consistency: FSCK and Journaling	557
41.1	A Detailed Example	558
41.2	Solution #1: The File System Checker	562
41.3	Solution #2: Journaling (or Write-Ahead Logging)	564
41.4	Solution #3: Other Approaches	574

41.5 Summary 575

References 576

42 Log-structured File Systems 579

42.1 Writing To Disk Sequentially 580

42.2 Writing Sequentially And Effectively 582

42.3 How Much To Buffer? 583

42.4 Finding Inodes 584

42.5 Solution Through Indirection: The Inode Map 585

42.6 The Checkpoint Region 586

42.7 Reading A File From Disk: A Recap 587

42.8 What About Directories? 588

42.9 A New Problem: Garbage Collection 589

42.10 Determining Block Liveness 591

42.11 A Policy Question: Which Blocks To Clean, And When? 592

42.12 Crash Recovery And The Log 592

42.13 Summary 593

References 595

43 Data Integrity and Protection 597

43.1 Disk Failure Modes 597

43.2 Handling Latent Sector Errors 600

43.3 Detecting Corruption: The Checksum 601

43.4 Using Checksums 604

43.5 A New Problem: Misdirected Writes 605

43.6 One Last Problem: Lost Writes 606

43.7 Scrubbing 607

43.8 Summary 607

References 608

44 Summary Dialogue on Persistence 611

45 A Dialogue on Distribution 613

46 Distributed Systems 615

46.1 Communication Basics 616

46.2 Unreliable Communication Layers 617

46.3 Reliable Communication Layers 620

46.4 Communication Abstractions 623

46.5 Remote Procedure Call (RPC) 625

46.6 Summary	630
References	632
47 Sun's Network File System (NFS)	633
47.1 A Basic Distributed File System	634
47.2 On To NFS	635
47.3 Focus: Simple and Fast Server Crash Recovery	635
47.4 Key To Fast Crash Recovery: Statelessness	636
47.5 The NFSv2 Protocol	638
47.6 From Protocol to Distributed File System	640
47.7 Handling Server Failure with Idempotent Operations	642
47.8 Improving Performance: Client-side Caching	644
47.9 The Cache Consistency Problem	645
47.10 Assessing NFS Cache Consistency	647
47.11 Implications on Server-Side Write Buffering	647
47.12 Summary	649
References	651
48 The Andrew File System (AFS)	653
48.1 AFS Version 1	653
48.2 Problems with Version 1	655
48.3 Improving the Protocol	656
48.4 AFS Version 2	657
48.5 Cache Consistency	658
48.6 Crash Recovery	658
48.7 Scale of AFSv2	659
48.8 Other Improvements: Namespaces, Security, Etc.	659
48.9 Summary	660
References	661
49 Summary Dialogue on Distribution	663
General Index	665
Asides	677
Tips	679
Cruces	683