

Chapters covered in ECE 3600

Contents

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

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
Homework 20

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 (Simulation)	35
5	Interlude: Process API	37
5.1	The <code>fork()</code> System Call	37
5.2	The <code>wait()</code> System Call	39
5.3	Finally, The <code>exec()</code> System Call	40
5.4	Why? Motivating The API	41
5.5	Process Control And Users	44
5.6	Useful Tools	45
5.7	Summary	45
	References	47
	Homework (Code)	48
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	55
6.4	Worried About Concurrency?	59
6.5	Summary	60
	References	62
	Homework (Measurement)	63
7	Scheduling: Introduction	65
7.1	Workload Assumptions	65
7.2	Scheduling Metrics	66
7.3	First In, First Out (FIFO)	66
7.4	Shortest Job First (SJF)	68
7.5	Shortest Time-to-Completion First (STCF)	69
7.6	A New Metric: Response Time	70
7.7	Round Robin	71
7.8	Incorporating I/O	73
7.9	No More Oracle	74
7.10	Summary	74
	References	75
	Homework (Simulation)	76
8	Scheduling: The Multi-Level Feedback Queue	77
8.1	MLFQ: Basic Rules	78
8.2	Attempt #1: How To Change Priority	79
8.3	Attempt #2: The Priority Boost	83
8.4	Attempt #3: Better Accounting	84
8.5	Tuning MLFQ And Other Issues	84
8.6	MLFQ: Summary	86
	References	87
	Homework (Simulation)	88

9	Scheduling: Proportional Share	89
9.1	Basic Concept: Tickets Represent Your Share	89
9.2	Ticket Mechanisms	91
9.3	Implementation	92
9.4	An Example	93
9.5	How To Assign Tickets?	94
9.6	Why Not Deterministic?	94
9.7	The Linux Completely Fair Scheduler (CFS)	95
9.8	Summary	100
	References	101
	Homework (Simulation)	102
10	Multiprocessor Scheduling (Advanced)	103
10.1	Background: Multiprocessor Architecture	104
10.2	Don't Forget Synchronization	106
10.3	One Final Issue: Cache Affinity	107
10.4	Single-Queue Scheduling	107
10.5	Multi-Queue Scheduling	109
10.6	Linux Multiprocessor Schedulers	112
10.7	Summary	112
	References	113
	Homework (Simulation)	114
11	Summary Dialogue on CPU Virtualization	117
12	A Dialogue on Memory Virtualization	119
13	The Abstraction: Address Spaces	121
13.1	Early Systems	121
13.2	Multiprogramming and Time Sharing	122
13.3	The Address Space	123
13.4	Goals	125
13.5	Summary	127
	References	128
	Homework (Code)	129
14	Interlude: Memory API	131
14.1	Types of Memory	131
14.2	The <code>malloc()</code> Call	132
14.3	The <code>free()</code> Call	134
14.4	Common Errors	134
14.5	Underlying OS Support	137
14.6	Other Calls	138
14.7	Summary	138
	References	139
	Homework (Code)	140

15 Mechanism: Address Translation	141
15.1 Assumptions	142
15.2 An Example	142
15.3 Dynamic (Hardware-based) Relocation	145
15.4 Hardware Support: A Summary	148
15.5 Operating System Issues	149
15.6 Summary	152
References	153
Homework (Simulation)	154
16 Segmentation	155
16.1 Segmentation: Generalized Base/Bounds	155
16.2 Which Segment Are We Referring To?	158
16.3 What About The Stack?	159
16.4 Support for Sharing	160
16.5 Fine-grained vs. Coarse-grained Segmentation	161
16.6 OS Support	161
16.7 Summary	163
References	164
Homework (Simulation)	165
17 Free-Space Management	167
17.1 Assumptions	168
17.2 Low-level Mechanisms	169
17.3 Basic Strategies	177
17.4 Other Approaches	179
17.5 Summary	181
References	182
Homework (Simulation)	183
18 Paging: Introduction	185
18.1 A Simple Example And Overview	185
18.2 Where Are Page Tables Stored?	189
18.3 What's Actually In The Page Table?	190
18.4 Paging: Also Too Slow	191
18.5 A Memory Trace	192
18.6 Summary	195
References	196
Homework (Simulation)	197
19 Paging: Faster Translations (TLBs)	199
19.1 TLB Basic Algorithm	199
19.2 Example: Accessing An Array	201
19.3 Who Handles The TLB Miss?	203
19.4 TLB Contents: What's In There?	205
19.5 TLB Issue: Context Switches	206
19.6 Issue: Replacement Policy	208

19.7	A Real TLB Entry	209
19.8	Summary	210
	References	211
	Homework (Measurement)	212
20	Paging: Smaller Tables	215
20.1	Simple Solution: Bigger Pages	215
20.2	Hybrid Approach: Paging and Segments	216
20.3	Multi-level Page Tables	219
20.4	Inverted Page Tables	226
20.5	Swapping the Page Tables to Disk	227
20.6	Summary	227
	References	228
	Homework (Simulation)	229
21	Beyond Physical Memory: Mechanisms	231
21.1	Swap Space	232
21.2	The Present Bit	233
21.3	The Page Fault	234
21.4	What If Memory Is Full?	235
21.5	Page Fault Control Flow	236
21.6	When Replacements Really Occur	237
21.7	Summary	238
	References	239
	Homework (Measurement)	240
22	Beyond Physical Memory: Policies	243
22.1	Cache Management	243
22.2	The Optimal Replacement Policy	244
22.3	A Simple Policy: FIFO	246
22.4	Another Simple Policy: Random	248
22.5	Using History: LRU	249
22.6	Workload Examples	250
22.7	Implementing Historical Algorithms	253
22.8	Approximating LRU	254
22.9	Considering Dirty Pages	255
22.10	Other VM Policies	256
22.11	Thrashing	256
22.12	Summary	257
	References	258
	Homework (Simulation)	259
23	Complete Virtual Memory Systems	261
23.1	VAX/VMS Virtual Memory	262
23.2	The Linux Virtual Memory System	268
23.3	Summary	277
	References	278

24	Summary Dialogue on Memory Virtualization	279
II Concurrency		283
25	A Dialogue on Concurrency	285
26	Concurrency: An Introduction	287
26.1	Why Use Threads?	288
26.2	An Example: Thread Creation	289
26.3	Why It Gets Worse: Shared Data	292
26.4	The Heart Of The Problem: Uncontrolled Scheduling	294
26.5	The Wish For Atomicity	296
26.6	One More Problem: Waiting For Another	298
26.7	Summary: Why in OS Class?	298
	References	300
	Homework (Simulation)	301
27	Interlude: Thread API	303
27.1	Thread Creation	303
27.2	Thread Completion	304
27.3	Locks	307
27.4	Condition Variables	309
27.5	Compiling and Running	311
27.6	Summary	311
	References	313
	Homework (Code)	314
28	Locks	315
28.1	Locks: The Basic Idea	315
28.2	Pthread Locks	316
28.3	Building A Lock	317
28.4	Evaluating Locks	317
28.5	Controlling Interrupts	318
28.6	A Failed Attempt: Just Using Loads/Stores	319
28.7	Building Working Spin Locks with Test-And-Set	320
28.8	Evaluating Spin Locks	322
28.9	Compare-And-Swap	323
28.10	Load-Linked and Store-Conditional	324
28.11	Fetch-And-Add	326
28.12	Too Much Spinning: What Now?	327
28.13	A Simple Approach: Just Yield, Baby	328
28.14	Using Queues: Sleeping Instead Of Spinning	329
28.15	Different OS, Different Support	332
28.16	Two-Phase Locks	332
28.17	Summary	334
	References	335

Homework (Simulation)	336
29 Lock-based Concurrent Data Structures	337
29.1 Concurrent Counters	337
29.2 Concurrent Linked Lists	342
29.3 Concurrent Queues	345
29.4 Concurrent Hash Table	346
29.5 Summary	348
References	349
Homework (Code)	350
30 Condition Variables	351
30.1 Definition and Routines	352
30.2 The Producer/Consumer (Bounded Buffer) Problem	355
30.3 Covering Conditions	363
30.4 Summary	364
References	365
Homework (Code)	366
31 Semaphores	367
31.1 Semaphores: A Definition	367
31.2 Binary Semaphores (Locks)	369
31.3 Semaphores For Ordering	370
31.4 The Producer/Consumer (Bounded Buffer) Problem	372
31.5 Reader-Writer Locks	376
31.6 The Dining Philosophers	378
31.7 How To Implement Semaphores	381
31.8 Summary	382
References	383
Homework (Code)	384
32 Common Concurrency Problems	385
32.1 What Types Of Bugs Exist?	385
32.2 Non-Deadlock Bugs	386
32.3 Deadlock Bugs	389
32.4 Summary	397
References	399
Homework (Code)	400
33 Event-based Concurrency (Advanced)	401
33.1 The Basic Idea: An Event Loop	401
33.2 An Important API: <code>select()</code> (or <code>poll()</code>)	402
33.3 Using <code>select()</code>	403
33.4 Why Simpler? No Locks Needed	404
33.5 A Problem: Blocking System Calls	405
33.6 A Solution: Asynchronous I/O	405
33.7 Another Problem: State Management	408

33.8	What Is Still Difficult With Events	409
33.9	Summary	409
	References	410
	Homework (Code)	411
34	Summary Dialogue on Concurrency	413
III Persistence		415
35	A Dialogue on Persistence	417
36	I/O Devices	419
36.1	System Architecture	419
36.2	A Canonical Device	421
36.3	The Canonical Protocol	422
36.4	Lowering CPU Overhead With Interrupts	423
36.5	More Efficient Data Movement With DMA	424
36.6	Methods Of Device Interaction	425
36.7	Fitting Into The OS: The Device Driver	426
36.8	Case Study: A Simple IDE Disk Driver	427
36.9	Historical Notes	430
36.10	Summary	430
	References	431
37	Hard Disk Drives	433
37.1	The Interface	433
37.2	Basic Geometry	434
37.3	A Simple Disk Drive	435
37.4	I/O Time: Doing The Math	438
37.5	Disk Scheduling	442
37.6	Summary	446
	References	447
	Homework (Simulation)	448
38	Redundant Arrays of Inexpensive Disks (RAIDs)	449
38.1	Interface And RAID Internals	450
38.2	Fault Model	451
38.3	How To Evaluate A RAID	451
38.4	RAID Level 0: Striping	452
38.5	RAID Level 1: Mirroring	455
38.6	RAID Level 4: Saving Space With Parity	458
38.7	RAID Level 5: Rotating Parity	462
38.8	RAID Comparison: A Summary	463
38.9	Other Interesting RAID Issues	464
38.10	Summary	464
	References	465

Homework (Simulation)	466
39 Interlude: Files and Directories	467
39.1 Files And Directories	467
39.2 The File System Interface	469
39.3 Creating Files	469
39.4 Reading And Writing Files	470
39.5 Reading And Writing, But Not Sequentially	472
39.6 Shared File Table Entries: <code>fork()</code> And <code>dup()</code>	475
39.7 Writing Immediately With <code>fsync()</code>	477
39.8 Renaming Files	478
39.9 Getting Information About Files	479
39.10 Removing Files	480
39.11 Making Directories	480
39.12 Reading Directories	481
39.13 Deleting Directories	482
39.14 Hard Links	482
39.15 Symbolic Links	484
39.16 Permission Bits And Access Control Lists	485
39.17 Making And Mounting A File System	488
39.18 Summary	490
References	491
Homework (Code)	492
40 File System Implementation	493
40.1 The Way To Think	493
40.2 Overall Organization	494
40.3 File Organization: The Inode	496
40.4 Directory Organization	501
40.5 Free Space Management	501
40.6 Access Paths: Reading and Writing	502
40.7 Caching and Buffering	506
40.8 Summary	508
References	509
Homework (Simulation)	510
41 Locality and The Fast File System	511
41.1 The Problem: Poor Performance	511
41.2 FFS: Disk Awareness Is The Solution	513
41.3 Organizing Structure: The Cylinder Group	513
41.4 Policies: How To Allocate Files and Directories	515
41.5 Measuring File Locality	517
41.6 The Large-File Exception	518
41.7 A Few Other Things About FFS	520
41.8 Summary	522
References	523
Homework (Simulation)	524

42	Crash Consistency: FSK and Journaling	525
42.1	A Detailed Example	526
42.2	Solution #1: The File System Checker	529
42.3	Solution #2: Journaling (or Write-Ahead Logging)	531
42.4	Solution #3: Other Approaches	541
42.5	Summary	542
	References	543
	Homework (Simulation)	545
43	Log-structured File Systems	547
43.1	Writing To Disk Sequentially	548
43.2	Writing Sequentially And Effectively	549
43.3	How Much To Buffer?	550
43.4	Problem: Finding Inodes	551
43.5	Solution Through Indirection: The Inode Map	551
43.6	Completing The Solution: The Checkpoint Region	553
43.7	Reading A File From Disk: A Recap	553
43.8	What About Directories?	554
43.9	A New Problem: Garbage Collection	555
43.10	Determining Block Liveness	556
43.11	A Policy Question: Which Blocks To Clean, And When?	557
43.12	Crash Recovery And The Log	558
43.13	Summary	558
	References	560
	Homework (Simulation)	561
44	Flash-based SSDs	563
44.1	Storing a Single Bit	563
44.2	From Bits to Banks/Planes	564
44.3	Basic Flash Operations	565
44.4	Flash Performance And Reliability	567
44.5	From Raw Flash to Flash-Based SSDs	568
44.6	FTL Organization: A Bad Approach	569
44.7	A Log-Structured FTL	570
44.8	Garbage Collection	572
44.9	Mapping Table Size	574
44.10	Wear Leveling	579
44.11	SSD Performance And Cost	579
44.12	Summary	581
	References	583
	Homework (Simulation)	585
45	Data Integrity and Protection	587
45.1	Disk Failure Modes	587
45.2	Handling Latent Sector Errors	589
45.3	Detecting Corruption: The Checksum	590
45.4	Using Checksums	593

45.5	A New Problem: Misdirected Writes	594
45.6	One Last Problem: Lost Writes	595
45.7	Scrubbing	595
45.8	Overheads Of Checksumming	596
45.9	Summary	596
	References	597
	Homework (Simulation)	598
	Homework (Code)	599
46	Summary Dialogue on Persistence	601
47	A Dialogue on Distribution	603
48	Distributed Systems	605
48.1	Communication Basics	606
48.2	Unreliable Communication Layers	607
48.3	Reliable Communication Layers	609
48.4	Communication Abstractions	611
48.5	Remote Procedure Call (RPC)	613
48.6	Summary	618
	References	619
	Homework (Code)	620
49	Sun's Network File System (NFS)	621
49.1	A Basic Distributed File System	622
49.2	On To NFS	623
49.3	Focus: Simple And Fast Server Crash Recovery	623
49.4	Key To Fast Crash Recovery: Statelessness	624
49.5	The NFSv2 Protocol	625
49.6	From Protocol To Distributed File System	627
49.7	Handling Server Failure With Idempotent Operations	629
49.8	Improving Performance: Client-side Caching	631
49.9	The Cache Consistency Problem	631
49.10	Assessing NFS Cache Consistency	633
49.11	Implications On Server-Side Write Buffering	633
49.12	Summary	635
	References	637
	Homework (Measurement)	638
50	The Andrew File System (AFS)	639
50.1	AFS Version 1	639
50.2	Problems with Version 1	641
50.3	Improving the Protocol	642
50.4	AFS Version 2	642
50.5	Cache Consistency	644
50.6	Crash Recovery	646
50.7	Scale And Performance Of AFSv2	646

50.8	AFS: Other Improvements	649
50.9	Summary	650
	References	651
	Homework (Simulation)	652
51	Summary Dialogue on Distribution	653
	General Index	655
	Asides	667
	Tips	671
	Cruces	675