Operating Systems
Principles and Practice
Second Edition

Thomas Anderson
University of Washington

Michael Dahlin
University of Texas at Austin and Google

Recursive Books
recursivebooks.com
## Contents

### Preface

**Introduction** 3

1.1 What Is An Operating System? 6

- Resource Sharing: Operating System as Referee 9
- Masking Limitations: Operating System as Illusionist 11
- Providing Common Services: Operating System as Glue 12
- Operating System Design Patterns 13

1.2 Operating System Evaluation 19

- Reliability and Availability 19
- Security 20
- Portability 21
- Performance 23
- Adoption 24
- Design Tradeoffs 25

1.3 Operating Systems: Past, Present, and Future 26

- Impact of Technology Trends 26
- Early Operating Systems 27
- Multi-User Operating Systems 28
- Time-Sharing Operating Systems 29
- Modern Operating Systems 30
- Future Operating Systems 32

Exercises 33

### Part I Kernels and Processes

**The Kernel Abstraction** 39

2.1 The Process Abstraction 43

2.2 Dual-Mode Operation 44

- Privileged Instructions 47
- Memory Protection 49
- Timer Interrupts 56

2.3 Types of Mode Transfer 57

- User to Kernel Mode 57
- Kernel to User Mode 61

2.4 Implementing Safe Mode Transfer 61

- Interrupt Vector Table 63
- Interrupt Stack 64
- Two Stacks per Process 65
- Interrupt Masking 67

- Hardware Support for Saving and Restoring Registers 68

2.5 Putting It All Together: x86 Mode Transfer 69
Synchronizing Access to Shared Objects 183

5.1 Challenges 187
   Race Conditions 187
   Atomic Operations 189
   Too Much Milk 190
   Discussion 193
   A Better Solution 194

5.2 Structuring Shared Objects 195
   Implementing Shared Objects 196
   Scope and Roadmap 198

5.3 Locks: Mutual Exclusion 198
   Locks: API and Properties 199
   Case Study: Thread-Safe Bounded Queue 201

5.4 Condition Variables: Waiting for a Change 205
   Condition Variable Definition 206
   Thread Life Cycle Revisited 211
   Case Study: Blocking Bounded Queue 213

5.5 Designing and Implementing Shared Objects 214
   High Level Methodology 216
   Implementation Best Practices 220
   Three Pitfalls 224

5.6 Three Case Studies 227
   Readers/ Writers Lock 227
   Synchronization Barriers 231
   FIFO Blocking Bounded Queue 236

5.7 Implementing Synchronization Objects 237
   Implementing Uniprocessor Locks by Disabling Interrupts 238
   Implementing Uniprocessor Queueing Locks 239
   Implementing Multiprocessor Spinlocks 240
   Implementing Multiprocessor Queueing Locks 240
   Case Study: Linux 2.6 Kernel Mutex Lock 244
   Implementing Condition Variables 246
   Implementing Application-level Synchronization 246

5.8 Semaphores Considered Harmful 248

5.9 Summary and Future Directions 253
   Historical Notes 254

Exercises 254

Multi-Object Synchronization 261

6.1 Multiprocessor Lock Performance 262
6.2 Lock Design Patterns 266
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Lock Contention</td>
<td>272</td>
</tr>
<tr>
<td>6.3</td>
<td>Fine-Grained Locking</td>
<td>266</td>
</tr>
<tr>
<td>6.3</td>
<td>Per-Processor Data Structures</td>
<td>268</td>
</tr>
<tr>
<td>6.3</td>
<td>Ownership Design Pattern</td>
<td>268</td>
</tr>
<tr>
<td>6.3</td>
<td>Staged Architecture</td>
<td>269</td>
</tr>
<tr>
<td>6.4</td>
<td>Multi-Object Atomicity</td>
<td>282</td>
</tr>
<tr>
<td>6.4</td>
<td>Careful Class Design</td>
<td>283</td>
</tr>
<tr>
<td>6.4</td>
<td>Acquire-All/Release-All</td>
<td>283</td>
</tr>
<tr>
<td>6.4</td>
<td>Two-Phase Locking</td>
<td>284</td>
</tr>
<tr>
<td>6.5</td>
<td>Deadlock</td>
<td>285</td>
</tr>
<tr>
<td>6.5</td>
<td>Deadlock vs. Starvation</td>
<td>288</td>
</tr>
<tr>
<td>6.5</td>
<td>Necessary Conditions for Deadlock</td>
<td>289</td>
</tr>
<tr>
<td>6.5</td>
<td>Preventing Deadlock</td>
<td>291</td>
</tr>
<tr>
<td>6.5</td>
<td>The Banker’s Algorithm for Avoiding Deadlock</td>
<td>294</td>
</tr>
<tr>
<td>6.5</td>
<td>Detecting and Recovering From Deadlocks</td>
<td>299</td>
</tr>
<tr>
<td>6.6</td>
<td>Non-Blocking Synchronization</td>
<td>305</td>
</tr>
<tr>
<td>6.7</td>
<td>Summary and Future Directions</td>
<td>307</td>
</tr>
<tr>
<td>Exercises</td>
<td></td>
<td>308</td>
</tr>
<tr>
<td>7</td>
<td>Scheduling</td>
<td>313</td>
</tr>
<tr>
<td>7.1</td>
<td>Uniprocessor Scheduling</td>
<td>316</td>
</tr>
<tr>
<td>7.1</td>
<td>First-In-First-Out (FIFO)</td>
<td>317</td>
</tr>
<tr>
<td>7.1</td>
<td>Shortest Job First (SJF)</td>
<td>318</td>
</tr>
<tr>
<td>7.1</td>
<td>Round Robin</td>
<td>320</td>
</tr>
<tr>
<td>7.1</td>
<td>Max-Min Fairness</td>
<td>323</td>
</tr>
<tr>
<td>7.1</td>
<td>Case Study: Multi-Level Feedback</td>
<td>326</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>328</td>
</tr>
<tr>
<td>7.2</td>
<td>Multiprocessor Scheduling</td>
<td>329</td>
</tr>
<tr>
<td>7.2</td>
<td>Scheduling Sequential Applications on Multiprocessors</td>
<td>329</td>
</tr>
<tr>
<td>7.2</td>
<td>Scheduling Parallel Applications</td>
<td>331</td>
</tr>
<tr>
<td>7.3</td>
<td>Energy-Aware Scheduling</td>
<td>337</td>
</tr>
<tr>
<td>7.4</td>
<td>Real-Time Scheduling</td>
<td>340</td>
</tr>
<tr>
<td>7.5</td>
<td>Queueing Theory</td>
<td>343</td>
</tr>
<tr>
<td>7.5</td>
<td>Definitions</td>
<td>343</td>
</tr>
<tr>
<td>7.5</td>
<td>Little’s Law</td>
<td>345</td>
</tr>
<tr>
<td>7.5</td>
<td>Response Time Versus Utilization</td>
<td>348</td>
</tr>
<tr>
<td>7.5</td>
<td>“What if?” Questions</td>
<td>355</td>
</tr>
<tr>
<td>7.5</td>
<td>Lessons</td>
<td>358</td>
</tr>
<tr>
<td>7.6</td>
<td>Overload Management</td>
<td>358</td>
</tr>
<tr>
<td>7.7</td>
<td>Case Study: Servers in a Data Center</td>
<td>361</td>
</tr>
<tr>
<td>7.8</td>
<td>Summary and Future Directions</td>
<td>362</td>
</tr>
<tr>
<td>Exercises</td>
<td></td>
<td>363</td>
</tr>
<tr>
<td>Part III</td>
<td>Memory Management</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Address Translation</td>
<td>371</td>
</tr>
</tbody>
</table>
8.1 Address Translation Concept 374
8.2 Towards Flexible Address Translation 376
   Segmented Memory 376
   Paged Memory 382
   Multi-Level Translation 385
   Portability 390
8.3 Towards Efficient Address Translation 392
   Translation Lookaside Buffers 393
   Superpages 396
   TLB Consistency 398
   Virtually Addressed Caches 402
   Physically Addressed Caches 403
8.4 Software Protection 405
   Single Language Operating Systems 406
   Language-Independent Software Fault Isolation 410
   Sandboxes Via Intermediate Code 413
8.5 Summary and Future Directions 414
Exercises 415

9 Caching and Virtual Memory 421
9.1 Cache Concept 425
9.2 Memory Hierarchy 427
9.3 When Caches Work and When They Do Not 429
   Working Set Model 430
   Zipf Model 433
9.4 Memory Cache Lookup 434
9.5 Replacement Policies 439
   Random 439
   First-In-First-Out (FIFO) 440
   Optimal Cache Replacement (MIN) 441
   Least Recently Used (LRU) 442
   Least Frequently Used (LFU) 443
   Belady's Anomaly 444
9.6 Case Study: Memory-Mapped Files 445
   Advantages 446
   Implementation 447
   Approximating LRU 451
9.7 Case Study: Virtual Memory 454
   Self-Paging 454
   Swapping 456
9.8 Summary and Future Directions 457
Exercises 458

10 Advanced Memory Management 465
10.1 Zero-Copy I/O 467
10.2 Virtual Machines 470
   Virtual Machine Page Tables 470
   Transparent Memory Compression 472
10.3 Fault Tolerance 475
Part IV Persistent Storage

11 File Systems: Introduction and Overview

11.1 The File System Abstraction

11.2 API

11.3 Software Layers

11.4 Summary and Future Directions

Exercises

12 Storage Devices

12.1 Magnetic Disk

12.2 Flash Storage

12.3 Summary and Future Directions

Exercises

13 Files and Directories

13.1 Implementation Overview

13.2 Directories: Naming Data

13.3 Files: Finding Data

13.4 Putting It All Together: File and Directory Access

13.5 Summary and Future Directions

Exercises

14 Reliable Storage

14.1 Transactions: Atomic Updates

Ad Hoc Approaches

The Transaction Abstraction

Implementing Transactions