

# 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](http://recursivebooks.com)

*Operating Systems: Principles and Practice (Second Edition)* by Thomas Anderson and Michael Dahlin

Copyright © Thomas Anderson and Michael Dahlin, 2011-2014.

ISBN 978-0-9856735-2-9

*Publisher:* Recursive Books, Ltd.  
<http://recursivebooks.com/>  
*Cover:* Reflection Lake, Mt. Ranier  
*Cover design:* Cameron Neat  
*Illustrations:* Cameron Neat  
*Copy editors:* Sandy Kaplan, Whitney Schmidt  
*Printer:* Lightning Source

SUGGESTIONS, COMMENTS, and ERRORS. We welcome suggestions, comments and error reports, by email to [suggestions@recursivebooks.com](mailto:suggestions@recursivebooks.com)

Notice of rights. All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form by any means — electronic, mechanical, photocopying, recording, or otherwise — without the prior written permission of the publisher. For information on getting permissions for reprints and excerpts, contact [permissions@recursivebooks.com](mailto:permissions@recursivebooks.com)

Notice of liability. The information in this book is distributed on an “As Is” basis, without warranty. Neither the authors nor Recursive Books shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information or instructions contained in this book or by the computer software and hardware products described in it.

Trademarks: Throughout this book trademarked names are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state we are using the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark. All trademarks or service marks are the property of their respective owners.

# Contents

## 1

<b>Preface</b>	<b>x</b>
<b>Introduction</b>	<b>3</b>
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 2

<b>Kernels and Processes</b>	
<b>The Kernel Abstraction</b>	<b>39</b>
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

2.6	Implementing Secure System Calls	74
2.7	Starting a New Process	77
2.8	Implementing Upcalls	79
2.9	Case Study: Booting an Operating System Kernel	83
2.10	Case Study: Virtual Machines	84
2.11	Summary and Future Directions	87
	Exercises	90

### 3

	<b>The Programming Interface</b>	<b>95</b>
3.1	Process Management	100
	Windows Process Management	101
	UNIX Process Management	102
3.2	Input/Output	107
3.3	Case Study: Implementing a Shell	110
3.4	Case Study: Interprocess Communication	113
	Producer-Consumer Communication	114
	Client-Server Communication	115
3.5	Operating System Structure	115
	Monolithic Kernels	117
	Microkernel	122
3.6	Summary and Future Directions	123
	Exercises	124

## Part II Concurrency

### 4

	<b>Concurrency and Threads</b>	<b>129</b>
4.1	Thread Use Cases	132
	Four Reasons to Use Threads	134
4.2	Thread Abstraction	136
	Running, Suspending, and Resuming Threads	136
	Why “Unpredictable Speed”?	138
4.3	Simple Thread API	141
	A Multi-Threaded Hello World	142
	Fork-Join Parallelism	143
4.4	Thread Data Structures and Life Cycle	145
	Per-Thread State and Thread Control Block (TCB)	146
	Shared State	148
4.5	Thread Life Cycle	148
4.6	Implementing Kernel Threads	153
	Creating a Thread	155
	Deleting a Thread	157
	Thread Context Switch	157
4.7	Combining Kernel Threads and Single-Threaded User Processes	164
4.8	Implementing Multi-Threaded Processes	165
	Implementing Multi-Threaded Processes Using Kernel Threads	166
	Implementing User-Level Threads Without Kernel Support	167

**5**

	Implementing User-Level Threads With Kernel Support	168
4.9	Alternative Abstractions	170
	Asynchronous I/O and Event-Driven Programming	171
	Data Parallel Programming	175
4.10	Summary and Future Directions	176
	Historical Notes	177
	Exercises	178

**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

**6****Multi-Object Synchronization 261**

6.1	Multiprocessor Lock Performance	262
6.2	Lock Design Patterns	266

	Fine-Grained Locking	266
	Per-Processor Data Structures	268
	Ownership Design Pattern	268
	Staged Architecture	269
6.3	Lock Contention	272
	MCS Locks	272
	Read-Copy-Update (RCU)	274
6.4	Multi-Object Atomicity	282
	Careful Class Design	283
	Acquire-All/Release-All	283
	Two-Phase Locking	284
6.5	Deadlock	285
	Deadlock vs. Starvation	288
	Necessary Conditions for Deadlock	289
	Preventing Deadlock	291
	The Banker's Algorithm for Avoiding Deadlock	294
	Detecting and Recovering From Deadlocks	299
6.6	Non-Blocking Synchronization	305
6.7	Summary and Future Directions	307
	Exercises	308

## 7

	<b>Scheduling</b>	<b>313</b>
7.1	Uniprocessor Scheduling	316
	First-In-First-Out (FIFO)	317
	Shortest Job First (SJF)	318
	Round Robin	320
	Max-Min Fairness	323
	Case Study: Multi-Level Feedback	326
	Summary	328
7.2	Multiprocessor Scheduling	329
	Scheduling Sequential Applications on Multiprocessors	329
	Scheduling Parallel Applications	331
7.3	Energy-Aware Scheduling	337
7.4	Real-Time Scheduling	340
7.5	Queueing Theory	343
	Definitions	343
	Little's Law	345
	Response Time Versus Utilization	348
	"What if?" Questions	355
	Lessons	358
7.6	Overload Management	358
7.7	Case Study: Servers in a Data Center	361
7.8	Summary and Future Directions	362
	Exercises	363

## Part III Memory Management

### 8

	Address Translation	371
--	---------------------	-----

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

	<b>Caching and Virtual Memory</b>	<b>421</b>
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

	<b>Advanced Memory Management</b>	<b>465</b>
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

Checkpoint and Restart	476
Recoverable Virtual Memory	478
Deterministic Debugging	479
10.4 Security	481
10.5 User-Level Memory Management	482
10.6 Summary and Future Directions	485
Exercises	485

## Part IV Persistent Storage

<b>11</b>	<b>File Systems: Introduction and Overview</b>	<b>491</b>
	11.1 The File System Abstraction	495
	11.2 API	502
	11.3 Software Layers	505
	API and Performance	506
	Device Drivers: Common Abstractions	508
	Device Access	508
	Putting It All Together: A Simple Disk Request	511
	11.4 Summary and Future Directions	511
	Exercises	513
<b>12</b>	<b>Storage Devices</b>	<b>517</b>
	12.1 Magnetic Disk	517
	Disk Access and Performance	520
	Case Study: Toshiba MK3254GSY	523
	Disk Scheduling	527
	12.2 Flash Storage	530
	12.3 Summary and Future Directions	536
	Exercises	540
<b>13</b>	<b>Files and Directories</b>	<b>545</b>
	13.1 Implementation Overview	546
	13.2 Directories: Naming Data	547
	13.3 Files: Finding Data	553
	FAT: Linked List	555
	FFS: Fixed Tree	558
	NTFS: Flexible Tree With Extents	568
	Copy-On-Write File Systems	573
	13.4 Putting It All Together: File and Directory Access	581
	13.5 Summary and Future Directions	583
	Exercises	584
<b>14</b>	<b>Reliable Storage</b>	<b>589</b>
	14.1 Transactions: Atomic Updates	592
	Ad Hoc Approaches	593
	The Transaction Abstraction	595
	Implementing Transactions	597



---

Transactions and File Systems	609
14.2 Error Detection and Correction	613
Storage Device Failures and Mitigation	613
RAID: Multi-Disk Redundancy for Error Correction	619
Software Integrity Checks	631
14.3 Summary and Future Directions	633
Exercises	635
<b>References</b>	<b>641</b>
<b>Glossary</b>	<b>651</b>
<b>Index</b>	<b>662</b>
<b>About the Authors</b>	<b>669</b>