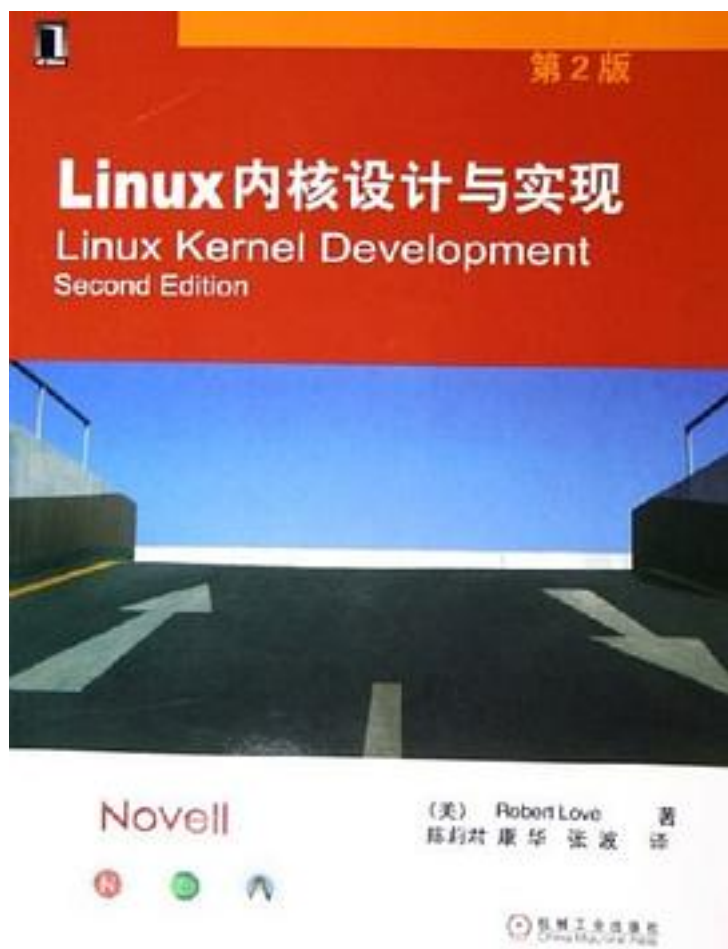


Linux内核设计与实现



[Linux内核设计与实现_下载链接1](#)

著者: (美) Robert Love

出版者: 机械工业出版社

出版时间: 2011-1

装帧:

isbn: 9787111327929

本书基于Linux 2.6内核介绍了Linux内核的设计与实现，涵盖了从核心内核系统的应用到内核设计与实现等各

方面内容，主要内容包括：进程管理、调度、时间管理和定时器、系统调用接口、内存寻址、内存管理、页缓

存、VFS、内核同步、可移植性、调试技术等。此外，本书还讨论了Linux 2.6颇具特色的内容，包括CFS调度

程序、抢占式内核、块I/O层以及I/O调度程序。

本书详细描述了Linux内核的主要子系统和特点，包括其设计、实现和接口，既介绍理论也讨论具体应用，填

补了Linux内核理论和实践细节之间的鸿沟，能够带领读者快速走进Linux内核世界，真正开发内核代码。

如果你是一名Linux内核爱好者，本书的内容可以帮助你大显身手。如果你是一名普通程序员，本书的内容将

会拓宽你的编程思路。如果你初次接触Linux内核，本书则可以帮助你对外核各个核心子系统有一个整体把握

。

本版新增内容

- 增加一章专门描述内核数据结构
- 详细描述中断处理程序
- 扩充虚拟内存和内存分配的内容
- 调试Linux内核的技巧
- 内核同步和锁机制的深度描述
- 提交内核补丁以及参与Linux内核社区的建设性建议

作者介绍:

Robert Love

是开源社区的名人，很早就开始使用Linux。目前他是Google公司高级软件工程师，是开发

Android移动平台内核的团队成员。他曾受聘于Novell公司，作为Linux Desktop主架构师。他还曾受聘于

MontaVista软件公司（后改名为Ximian公司），作为内核工程师。他的内核项目包括抢占式内核、进程调度程

序、内核事件层、inotify、VM增强以及设备驱动程序。他是《Linux Journal》杂志的特邀编辑。

目录: 1 Introduction to the Linux Kernel 1

History of Unix	1
Along Came Linus: Introduction to Linux	3
Overview of Operating Systems and Kernels	4
Linux Versus Classic Unix Kernels	6
Linux Kernel Versions	8
The Linux Kernel Development Community	10
Before We Begin	10
2 Getting Started with the Kernel	11
Obtaining the Kernel Source	11
Using Git	11
Installing the Kernel Source	12
Using Patches	12
The Kernel Source Tree	12
Building the Kernel	13
Configuring the Kernel	14
Minimizing Build Noise	15
Spawning Multiple Build Jobs	16
Installing the New Kernel	16
A Beast of a Different Nature	16
No libc or Standard Headers	17
GNU C	18
Inline Functions	18
Inline Assembly	19
Branch Annotation	19
No Memory Protection	20
No (Easy) Use of Floating Point	20
Small, Fixed-Size Stack	20
Synchronization and Concurrency	21
Importance of Portability	21
Conclusion	21
3 Process Management	23
The Process	23
Process Descriptor and the Task Structure	24
Allocating the Process Descriptor	25
Storing the Process Descriptor	26
Process State	27
Manipulating the Current Process State	29
Process Context	29
The Process Family Tree	29
Process Creation	31
Copy-on-Write	31
Forking	32
vfork()	33
The Linux Implementation of Threads	33
Creating Threads	34
Kernel Threads	35
Process Termination	36
Removing the Process Descriptor	37
The Dilemma of the Parentless Task	38
Conclusion	40
4 Process Scheduling	41
Multitasking	41
Linux's Process Scheduler	42
Policy	43

I/O-Bound Versus Processor-Bound Processes 43
Process Priority 44
Timeslice 45
The Scheduling Policy in Action 45
The Linux Scheduling Algorithm 46
Scheduler Classes 46
Process Scheduling in Unix Systems 47
Fair Scheduling 48
The Linux Scheduling Implementation 50
Time Accounting 50
The Scheduler Entity Structure 50
The Virtual Runtime 51
Process Selection 52
Picking the Next Task 53
Adding Processes to the Tree 54
Removing Processes from the Tree 56
The Scheduler Entry Point 57
Sleeping and Waking Up 58
Wait Queues 58
Waking Up 61
Preemption and Context Switching 62
User Preemption 62
Kernel Preemption 63
Real-Time Scheduling Policies 64
Scheduler-Related System Calls 65
Scheduling Policy and Priority-Related System Calls 66
Processor Affinity System Calls 66
Yielding Processor Time 66
Conclusion 67
5 System Calls 69
Communicating with the Kernel 69
APIs, POSIX, and the C Library 70
Syscalls 71
System Call Numbers 72
System Call Performance 72
System Call Handler 73
Denoting the Correct System Call 73
Parameter Passing 74
System Call Implementation 74
Implementing System Calls 74
Verifying the Parameters 75
System Call Context 78
Final Steps in Binding a System Call 79
Accessing the System Call from User-Space 81
Why Not to Implement a System Call 82
Conclusion 83
6 Kernel Data Structures 85
Linked Lists 85
Singly and Doubly Linked Lists 85
Circular Linked Lists 86
Moving Through a Linked List 87
The Linux Kernel's Implementation 88
The Linked List Structure 88

- Defining a Linked List 89
- List Heads 90
- Manipulating Linked Lists 90
- Adding a Node to a Linked List 90
- Deleting a Node from a Linked List 91
- Moving and Splicing Linked List Nodes 92
- Traversing Linked Lists 93
- The Basic Approach 93
- The Usable Approach 93
- Iterating Through a List Backward 94
- Iterating While Removing 95
- Other Linked List Methods 96
- Queues 96
- kfifo 97
- Creating a Queue 97
- Enqueuing Data 98
- Dequeuing Data 98
- Obtaining the Size of a Queue 98
- Resetting and Destroying the Queue 99
- Example Queue Usage 99
- Maps 100
- Initializing an idr 101
- Allocating a New UID 101
- Looking Up a UID 102
- Removing a UID 103
- Destroying an idr 103
- Binary Trees 103
- Binary Search Trees 104
- Self-Balancing Binary Search Trees 105
- Red-Black Trees 105
- rbtrees 106
- What Data Structure to Use, When 108
- Algorithmic Complexity 109
- Algorithms 109
- Big-O Notation 109
- Big Theta Notation 109
- Time Complexity 110
- Conclusion 111
- 7 Interrupts and Interrupt Handlers 113
- Interrupts 113
- Interrupt Handlers 114
- Top Halves Versus Bottom Halves 115
- Registering an Interrupt Handler 116
- Interrupt Handler Flags 116
- An Interrupt Example 117
- Freeing an Interrupt Handler 118
- Writing an Interrupt Handler 118
- Shared Handlers 119
- A Real-Life Interrupt Handler 120
- Interrupt Context 122
- Implementing Interrupt Handlers 123
- /proc/interrupts 126
- Interrupt Control 127
- Disabling and Enabling Interrupts 127

Disabling a Specific Interrupt Line 129
Status of the Interrupt System 130
Conclusion 131
8 Bottom Halves and Deferring Work 133
Bottom Halves 134
Why Bottom Halves? 134
A World of Bottom Halves 135
The Original “Bottom Half” 135
Task Queues 135
Softirqs and Tasklets 136
Dispelling the Confusion 137
Softirqs 137
Implementing Softirqs 137
The Softirq Handler 138
Executing Softirqs 138
Using Softirqs 140
Assigning an Index 140
Registering Your Handler 141
Raising Your Softirq 141
Tasklets 142
Implementing Tasklets 142
The Tasklet Structure 142
Scheduling Tasklets 143
Using Tasklets 144
Declaring Your Tasklet 144
Writing Your Tasklet Handler 145
Scheduling Your Tasklet 145
ksoftirqd 146
The Old BH Mechanism 148
Work Queues 149
Implementing Work Queues 149
Data Structures Representing the Threads 149
Data Structures Representing the Work 150
Work Queue Implementation Summary 152
Using Work Queues 153
Creating Work 153
Your Work Queue Handler 153
Scheduling Work 153
Flushing Work 154
Creating New Work Queues 154
The Old Task Queue Mechanism 155
Which Bottom Half Should I Use? 156
Locking Between the Bottom Halves 157
Disabling Bottom Halves 157
Conclusion 159
9 An Introduction to Kernel Synchronization 161
Critical Regions and Race Conditions 162
Why Do We Need Protection? 162
The Single Variable 163
Locking 165
Causes of Concurrency 167
Knowing What to Protect 168
Deadlocks 169
Contention and Scalability 171

Conclusion 172
10 Kernel Synchronization Methods 175
Atomic Operations 175
Atomic Integer Operations 176
64-Bit Atomic Operations 180
Atomic Bitwise Operations 181
Spin Locks 183
Spin Lock Methods 184
Other Spin Lock Methods 186
Spin Locks and Bottom Halves 187
Reader-Writer Spin Locks 188
Semaphores 190
Counting and Binary Semaphores 191
Creating and Initializing Semaphores 192
Using Semaphores 193
Reader-Writer Semaphores 194
Mutexes 195
Semaphores Versus Mutexes 197
Spin Locks Versus Mutexes 197
Completion Variables 197
BKL: The Big Kernel Lock 198
Sequential Locks 200
Preemption Disabling 201
Ordering and Barriers 203
Conclusion 206
11 Timers and Time Management 207
Kernel Notion of Time 208
The Tick Rate: HZ 208
The Ideal HZ Value 210
Advantages with a Larger HZ 210
Disadvantages with a Larger HZ 211
Jiffies 212
Internal Representation of Jiffies 213
Jiffies Wraparound 214
User-Space and HZ 216
Hardware Clocks and Timers 216
Real-Time Clock 217
System Timer 217
The Timer Interrupt Handler 217
The Time of Day 220
Timers 222
Using Timers 222
Timer Race Conditions 224
Timer Implementation 224
Delaying Execution 225
Busy Looping 225
Small Delays 226
`schedule_timeout()` 227
`schedule_timeout()` Implementation 228
Sleeping on a Wait Queue, with a Timeout 229
Conclusion 230
12 Memory Management 231
Pages 231
Zones 233

- Getting Pages 235
- Getting Zeroed Pages 236
- Freeing Pages 237
- kmalloc() 238
- gfp_mask Flags 238
- Action Modifiers 239
- Zone Modifiers 240
- Type Flags 241
- kfree() 243
- vmalloc() 244
- Slab Layer 245
 - Design of the Slab Layer 246
 - Slab Allocator Interface 249
 - Allocating from the Cache 250
 - Example of Using the Slab Allocator 251
 - Statically Allocating on the Stack 252
 - Single-Page Kernel Stacks 252
 - Playing Fair on the Stack 253
 - High Memory Mappings 253
 - Permanent Mappings 254
 - Temporary Mappings 254
 - Per-CPU Allocations 255
 - The New percpu Interface 256
 - Per-CPU Data at Compile-Time 256
 - Per-CPU Data at Runtime 257
 - Reasons for Using Per-CPU Data 258
 - Picking an Allocation Method 259
 - Conclusion 260
- 13 The Virtual Filesystem 261
 - Common Filesystem Interface 261
 - Filesystem Abstraction Layer 262
 - Unix Filesystems 263
 - VFS Objects and Their Data Structures 265
 - The Superblock Object 266
 - Superblock Operations 267
 - The Inode Object 270
 - Inode Operations 271
 - The Dentry Object 275
 - Dentry State 276
 - The Dentry Cache 276
 - Dentry Operations 278
 - The File Object 279
 - File Operations 280
 - Data Structures Associated with Filesystems 285
 - Data Structures Associated with a Process 286
 - Conclusion 288
- 14 The Block I/O Layer 289
 - Anatomy of a Block Device 290
 - Buffers and Buffer Heads 291
 - The bio Structure 294
 - I/O vectors 295
 - The Old Versus the New 296
 - Request Queues 297
 - I/O Schedulers 297

- The Job of an I/O Scheduler 298
- The Linus Elevator 299
- The Deadline I/O Scheduler 300
- The Anticipatory I/O Scheduler 302
- The Complete Fair Queuing I/O Scheduler 303
- The Noop I/O Scheduler 303
- I/O Scheduler Selection 304
- Conclusion 304
- 15 The Process Address Space 305
 - Address Spaces 305
 - The Memory Descriptor 306
 - Allocating a Memory Descriptor 308
 - Destroying a Memory Descriptor 309
 - The mm_struct and Kernel Threads 309
 - Virtual Memory Areas 309
 - VMA Flags 311
 - VMA Operations 312
 - Lists and Trees of Memory Areas 313
 - Memory Areas in Real Life 314
 - Manipulating Memory Areas 315
 - find_vma() 316
 - find_vma_prev() 317
 - find_vma_intersection() 317
 - mmap() and do_mmap(): Creating an Address Interval 318
 - munmap() and do_munmap(): Removing an Address Interval 320
 - Page Tables 320
 - Conclusion 322
- 16 The Page Cache and Page Writeback 323
 - Approaches to Caching 323
 - Write Caching 324
 - Cache Eviction 324
 - Least Recently Used 325
 - The Two-List Strategy 325
 - The Linux Page Cache 326
 - The address_space Object 326
 - address_space Operations 328
 - Radix Tree 330
 - The Old Page Hash Table 330
 - The Buffer Cache 330
 - The Flusher Threads 331
 - Laptop Mode 333
 - History: bdflush, kupdated, and pdflush 333
 - Avoiding Congestion with Multiple Threads 334
 - Conclusion 335
- 17 Devices and Modules 337
 - Device Types 337
 - Modules 338
 - Hello, World! 338
 - Building Modules 340
 - Living in the Source Tree 340
 - Living Externally 342
 - Installing Modules 342

- Generating Module Dependencies 342
- Loading Modules 343
- Managing Configuration Options 344
- Module Parameters 346
- Exported Symbols 348
- The Device Model 348
- Kobjects 349
- Ktypes 350
- Ksets 351
- Interrelation of Kobjects, Ktypes, and Ksets 351
- Managing and Manipulating Kobjects 352
- Reference Counts 353
- Incrementing and Decrementing Reference Counts 354
- Krefs 354
- sysfs 355
- Adding and Removing kobjects from sysfs 357
- Adding Files to sysfs 358
- Default Attributes 358
- Creating New Attributes 359
- Destroying Attributes 360
- sysfs Conventions 360
- The Kernel Events Layer 361
- Conclusion 362
- 18 Debugging 363
- Getting Started 363
- Bugs in the Kernel 364
- Debugging by Printing 364
- Robustness 365
- Loglevels 365
- The Log Buffer 366
- syslogd and klogd 367
- Transposing printf() and printk() 367
- Oops 367
- ksymoops 369
- kallsyms 369
- Kernel Debugging Options 370
- Asserting Bugs and Dumping Information 370
- Magic SysRq Key 371
- The Saga of a Kernel Debugger 372
- gdb 372
- kgdb 373
- Poking and Probing the System 373
- Using UID as a Conditional 373
- Using Condition Variables 374
- Using Statistics 374
- Rate and Occurrence Limiting Your Debugging 375
- Binary Searching to Find the Culprit Change 376
- Binary Searching with Git 376
- When All Else Fails: The Community 377
- Conclusion 378
- 19 Portability 379
- Portable Operating Systems 379
- History of Portability in Linux 380

Word Size and Data Types 381
Opaque Types 384
Special Types 384
Explicitly Sized Types 385
Signedness of Chars 386
Data Alignment 386
Avoiding Alignment Issues 387
Alignment of Nonstandard Types 387
Structure Padding 387
Byte Order 389
Time 391
Page Size 391
Processor Ordering 392
SMP, Kernel Preemption, and High Memory 393
Conclusion 393
20 Patches, Hacking, and the Community 395
The Community 395
Linux Coding Style 396
Indentation 396
Switch Statements 396
Spacing 397
Braces 398
Line Length 399
Naming 400
Functions 400
Comments 400
Typedefs 401
Use Existing Routines 402
Minimize ifdefs in the Source 402
Structure Initializers 402
Fixing Up Code Ex Post Facto 403
Chain of Command 403
Submitting Bug Reports 403
Patches 404
Generating Patches 404
Generating Patches with Git 405
Submitting Patches 406
Conclusion 406
Bibliography 407
Index 411
• • • • • (收起)

[Linux内核设计与实现 下载链接1](#)

标签

linux

操作系统

内核

kernel

Linux

计算机

原版书

Linux/Unix

评论

这本书经常掉书袋，用一些故作高深的词汇，像p42的 *raison d'etre*，还好我学过一点法语，尼玛我居然看懂了。

现在看来还是太浅了，适合入门。

应该是看完了…… 各种 struct ……

LKD

被逼无奈看了英文版，有个别小错误，但是作者功力真是深厚啊，这么薄的书讲的这么多，非常有帮助！

终于算是囫圇吞枣的读完一遍了，还是有很多地方不明白，有些章节写的稍微细了点，对于内核初学者来说反而不好

还是不太懂

稍浅，对于入门挺好的，但是有些地方介绍的太简单，反而不利于理解了。

太简略

虽然还是不错的，不过似乎有些部分已经落后了点，和当前的实现颇有些差异了。

我是有多脑残会突然想去读编程？

不错。

读的第一本英文原版书

Linux系统概要设计

love linux, love life.

内容比较浅，主要是介绍。觉得更适合给写应用的人看。

当时好像没有完全读下来吧？

言简意赅。

内核综述，前后串联很好，尤其对存储的管理阐述很条理。假期看第二遍，要结合源代码开始看，并且补看最后3章。可以配合ULK看，不过ULK中一些内容较老，所以源码才是最终依据。

大约一年半前泛读过一遍（当时看的是打印版），给人的印象是一本标准的OS教材。目前准备跟着读书小组边学习边看再来一遍 --

[Linux内核设计与实现 下载链接1](#)

书评

LDK这书估计慕名而来的人都会第一时间略感失望，首先书很薄，而且讲解不求深入。如果一个人在第一次翻阅此书的时候有这样的印象，那应该好好反省下自己是否太浮躁了。其实这部书的定位有点不高不低，但也正因如此，它是最适合过渡阶段的内核学习者阅读的一部书。正确的阅读...

Robert Love是个传奇人物。
传奇的原因是，当他还是大四学生的时候，已经有了7年的linux经验，并设计了linux的抢占式内核——2.4到2.6版内核的最关键进步之一。现在找到这个传奇在中国流传的源头，是一篇2002年初题为《看看国外的本科生能做什么？》的对当时大四的Robert Love...

能够把linux内核在短短300页叙述一遍，本身就是高难度的事情。但这本书确实做到了。

这本书基本是在俯视linux内核。全书很少涉及具体实现，而是把握思想，讲解算法，可以了解到linux内核的大概，而不用纠缠于具体细节。
而且这本书虽然使用的最新2.6版内核做讲解，但穿插历史...

自己一开始看的时候，觉得有些上下文提到的概念没有解释得很清楚，如果原来没有这方面的知识就会有一些困难。我自己是同时参考下面两本书一起看的。Understanding Linux Kernel 3rd Unix Internals 发现不懂就去查查上面两本书。这样基本都能看懂了。

我是对照中英文看的，去买了本译本，下了英文的ebook，主要是还是想赶点时间出来。中文的译文文笔倒还不错，至少很多笑话翻译得非常恰当，呵呵。但是致命伤也不少：
第一，排版上问题很大。很多原来的粗体斜体对关键词的标识根本就消失了。译者有时候弄点译者注，竟然就直接在...

此书已二读。2016年6月18日。窃以为此书不仅可以入门，还可入迷。全书虽然言简但的确意赅。设计方面的东西讲了很多，细节你就rtfc罢。要是再有点图就五星了。关于子系统的划分也很好，为后面的书打下良好之基础。另外由于是抢占式内核的设计者，关于抢占的说法也非常权威。泱泱...

我对作者写作意图的理解是：作者希望读者看了这本书之后，能够知道怎么运用内核函数来开发（驱动程序），也就是本书的书名，kernel development（中文翻成了设计与实现，但是请仔细体会一下，development和设计与实现并不是一回事）。基于这个目的，作者不纠结于内核具体的实现...

P138 注释1 幸好Linux没有提供这样的递归锁。【Windows下的Mutex和Critical Section是可递归的。Linux下的pthread_mutex_t锁默认是非递归的。可以显示的设置PTHREAD_MUTEX_RECURSIVE属性，将pthread_mutex_t设为递归锁。<http://fwd4.me/0AeU>】

因为对操作系统有些疑惑，就读了一下这本书。
这本书并不是专门讲操作系统理论的，而是专注阐释linux内核的机制和相应的实现方法，包括进程管理调度、内存管理、内核同步、定时器、文件系统等。
同时本书也附有内核源码用以帮助读者理解，读完之后确实消除了自己对...

作者的功力相当深厚，提纲挈领的介绍了内核的方方面面，而没有纠缠于细节，但又有细节介绍（比如O1调度器等），作为入门书最好不过了。因为ULK特别像一个手册，逻辑性不强，如果直接看，很容易陷入细节无法出来。如果先看这边书再去看ULK（和内核代码）就能很有针对性了。现在...

1、china-pub新浪微博免费赠书（5本）

#china-pub赠书#共5册，《云计算核心技术剖析》<http://t.cn/hehwpJ>《云计算(第二版)》
<http://t.cn/he3uWG>《Linux内核设计与实现(原书第3版)》<http://t.cn/aKbpeg>《黑客与画家:硅谷创业之父Paul Graham文集》<http://t.cn/hdhFN1>《Mongo...

先是看了一下电子版 觉得不错 于是在china-pub上买了这书（相比较电子版纸书做笔记比较方便 自己读书的习惯：）） 如果一上来就看understanding the linux kernel 3rd Edition 未免太过吃力 要是先仔细读完这本书在看前者 就容易的多 不算厚的篇幅把kernel大体上讲了一遍 ...

对于这本加上目录、附录共400页的小书本（相对于ULK那本大砖头...）我们还能要求什么更多的呢。
对于一个对内核感兴趣但是又无从下手的人来说我推荐读这一本，说内核的书籍确实很多，但是我觉得讲活得却不多。这本书给读者一个很好的框架。
简洁、点到为止，就好。如果想有...

在此奉上我学习LKD第三版的导图笔记，我所参考的linux源码版本是3.16。所有章节将逐步补充完整，欢迎大家与我硬核讨论。__^^__ 第3章 进程管理 第4章 进程调度 第5章 系统调用 [<https://www.edrawsoft.cn/viewer/public/s/b9150540150310>] 第7章 中断及中断handler [<https://ww...>]

提纲挈领，对内核重点的把握相当的精准到位！一本不可多得的从工程角度来讲解内核的书籍！需要有一定的linux内核实践功底！不建议作为入门的书来读，会很吃力！这本书可以看做是深入理解linux内核的笔记！

写的想当不错，深入浅出，把握住大方向，又不失细节，更重要的是能告诉你为什么这样做了，背后的目的是什么，而且每个章节很连贯，一章内容看似很多，如果认真看，看着看着一章就完了，ulk写得像字典似的，不容易看懂，开始建议看这本书。

我作为Linux内核学习的入门书来读的，基本上达到了我的目的。让读者能从一个总览式的视角大体了解了一下Linux内核。

就写出来的内容来说作者基本上做到了通俗易懂，但问题就在于对于Linux内核这么复杂的系统，作者介绍的并不够，让人有种迷茫的感觉。我想这本书也应该读两遍，...

在读这本书得时候，我把本科的操作系统和linux的命令忘得所剩无几，直接在昏暗的屋子里看源码和《深入理解linux内核》这本书的时候，心都要碎了。

陷入了只见树木，不见森林。后来在知乎上，看见很多人都推荐这本LKD就买来看。思路比较清晰、易读。像给了一面地图...

看了若干页，网上的试读，硬伤还是不少：===== p3

注二：“内核代码树种”，植物学家？！ p4 正文：“系统调用界面”，有点不专业！

p5 正文：“空进程”，idle进程好吧？！这个是专有名称了，别瞎改！

正文：“monolithic static binary”翻译成了“不可分割的静...

[Linux内核设计与实现_下载链接1](#)