HZ BOOKS

经　典　原　版　书　库

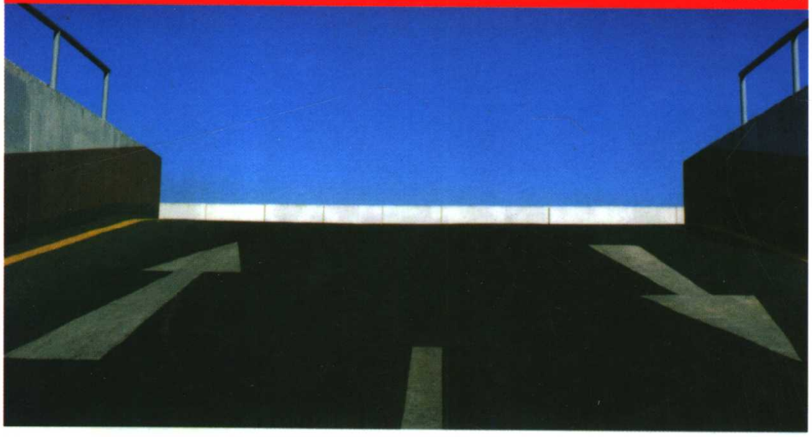
Linux内核设计与实现

(英文版 · 第2版)

DEVELOPMENT

Linux Kernel Development
Second Edition

NOVELL PRESS



Robert Love

Senior Kernel Engineer
Ximian Desktop Group, Novell, Inc.

PEARSON Education

Novell.

(美) Robert Love 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

Linux内核设计与实现

(英文版 · 第2版)

Linux Kernel Development

(Second Edition)

(美) Robert Love 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2005-4834

图书在版编目（CIP）数据

Linux内核设计与实现（英文版 · 第2版）/（美）洛夫（Love, R.）著. –北京：机械工业出版社，2006.1
（经典原版书库）
书名原文：Linux Kernel Development, Second Edition
ISBN 7-111-17575-1

Ⅰ. L…　Ⅱ. 洛…　Ⅲ. Linux操作系统–程序设计–英文　Ⅳ. TP316.89

中国版本图书馆CIP数据核字（2005）第119347号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春
北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行
2006年1月第1版第1次印刷
718mm × 1020mm 1/16 · 26.75印张
印数：0 001-3 000册
定价：40.00元

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔

滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

Praise for the first edition of Robert Love's *Linux Kernel Development*

"*Linux Kernel Development* is a starting point that will prove useful to any developer trying to get up to speed with one or more kernel subsystems.... The presentation is clear, the writing is enjoyable to read, and the information is accurate and useful. *Linux Kernel Development* belongs on the shelf of any developer who is interested in kernel work."

—Jonathan Corbet, *LWN.net*

"Developers who want to jump in and help out the greater cause, will find this book the book to have.... If you like the challenge, or if you need to know more about the Linux kernel in general for any reason, this is definitely the book to get. It would certainly be one of your best reads on operating systems topics."

—Eugenia Loli-Queru, *OSNews*

"If you are a Linux wizard or want to become one, you'll need *Linux Kernel Development* by Robert Love. It starts with a brief but useful introduction to operating systems and kernels and then goes on to break down the architecture of Linux very clearly. It also has an attribute we love: It gets to the point quickly, making the reasonable assumption that the reader has a clue."

—Mark Gibbs, *NetworkWorld*

"The author's knowledge, passion and excitement about the kernel show greatly and make this an enjoyable, easy-to-follow and compelling read.... I've already applied some of the knowledge in my normal application development on Linux at work.... If you're looking to do some kernel hacking or just interested in how the Linux kernel works, then I'd recommend reading this book.

—Richard Dawe

"One of the very best technical books I own.... The author presents information the way I want to see it. I really like the historical background information on Linux and comparisons to Unix. If you are looking for one single reference on Linux programming, I recommend this title. Not just for kernel hacking."

—Tom Gross, *practical.org*

Foreword

As the Linux kernel and the applications that use it become more widely used, we are seeing an increasing number of system software developers who wish to become involved in the development and maintenance of Linux. Some of these engineers are motivated purely by personal interest, some work for Linux companies, some work for hardware manufacturers, and some are involved with in-house development projects.

But all face a common problem: The learning curve for the kernel is getting longer and steeper. The system is becoming increasingly complex, and it is very large. And as the years pass, the current members of the kernel development team gain deeper and broader knowledge of the kernel's internals, which widens the gap between them and newcomers.

I believe that this declining accessibility of the Linux source base is already a problem for the quality of the kernel, and it will become more serious over time. Those who care for Linux clearly have an interest in increasing the number of developers who can contribute to the kernel.

One approach to this problem is to keep the code clean: sensible interfaces, consistent layout, "do one thing, do it well," and so on. This is Linus Torvalds' solution.

The approach that I counsel is to liberally apply commentary to the code: words that the reader can use to understand what the coder intended to achieve at the time. (The process of identifying divergences between the intent and the implementation is known as debugging. It is hard to do this if the intent is not known.)

But even code commentary does not provide the broad-sweep view of what a major subsystem is intended to do, and how its developers set about doing it.

This, the starting point of understanding, is what the written word serves best.

Robert Love's contribution provides a means by which experienced developers can gain that essential view of what services the kernel subsystems are supposed to provide, and how they set about providing them. This will be sufficient knowledge for many people: the curious, the application developers, those who wish to evaluate the kernel's design, and others.

But the book is also a stepping stone to take aspiring kernel developers to the next stage, which is making alterations to the kernel to achieve some defined objective. I would encourage aspiring developers to get their hands dirty: The best way to understand a part of the kernel is to make changes to it. Making a change forces the developer to a level of understanding that merely reading the code does not provide. The serious kernel developer will join the development mailing lists and will interact with other developers. This is the primary means by which kernel contributors learn and stay abreast. Robert covers the mechanics and culture of this important part of kernel life well.

Please enjoy and learn from Robert's book. And should you decide to take the next step and become a member of the kernel development community, consider yourself welcomed in advance. We value and measure people by the usefulness of their contributions, and when you contribute to Linux, you do so in the knowledge that your work is of small but immediate benefit to tens or even hundreds of millions of human beings. This is a most enjoyable privilege and responsibility.

Andrew Morton
Open Source Development Labs

Preface

When I was first approached about converting my experiences with the Linux kernel into a book, I proceeded with trepidation. I did not want to write simply yet another kernel book. Sure, there are not *that many* books on the subject, but I still wanted my approach to be somehow unique. What would place my book at the top of its subject? I was not motivated unless I could do something special, a best-in-class work.

I then realized that I could offer quite a unique approach to the topic. My job is hacking the kernel. My hobby is hacking the kernel. My love is hacking the kernel. Over the years, I have surely accumulated interesting anecdotes and important tips. With my experiences, I could write a book on how to hack the kernel and—more importantly—how *not* to hack the kernel. Primarily, this is a book about the design and implementation of the Linux kernel. The book's approach differs from would-be competition, however, in that the information is given with a slant to learning enough to actually get work done—and getting it done right. I am a pragmatic guy and this is a practical book. It should be fun, easy to read, and useful.

I hope that readers can walk away from this book with a better understanding of the rules (written and unwritten) of the kernel. I hope readers, fresh from reading this book and the kernel source code, can jump in and start writing useful, correct, clean kernel code. Of course, you can read this book just for fun, too.

That was the first edition. Time has passed, and now we return once more to the fray. This edition offers quite a bit over the first: intense polish and revision, updates, and many fresh sections and all new chapters. Changes in the kernel since the first edition have been recognized. More importantly, however, is the decision made by the Linux kernel community[1] to not proceed with a 2.7 development kernel in the near feature. Instead, kernel developers plan to continue developing and stabilizing 2.6. This implies many things, but one big item of relevance to this book is that there is quite a bit of staying power in a recent book on the 2.6 Linux kernel. If things do not move too quickly, there is a greater chance of a captured snapshot of the kernel remaining relevant long into the future. A book can finally rise up and become the canonical documentation for the kernel. I hope that you are holding that book.

Anyhow, here it is. I hope you enjoy it.

1 This decision was made in the summer of 2004 at the annual Linux Kernel Developers Summit in Ottawa, Canada.

So Here We Are

Developing code in the kernel does not require genius, magic, or a bushy Unix-hacker beard. The kernel, although having some interesting rules of its own, is not much different from any other large software endeavor. There is much to learn—as with any big project—but there is not too much about the kernel that is more sacred or confusing than anything else.

It is imperative that you utilize the source. The open availability of the source code for the Linux system is a rarity that we must not take for granted. It is not sufficient *only* to read the source, however. You need to dig in and change some code. Find a bug and fix it. Improve the drivers for your hardware. Find an itch and scratch it! Only when you *write* code will it all come together.

Kernel Version

This book is based on the 2.6 Linux kernel series. Specifically, it is up to date as of Linux kernel version 2.6.10. The kernel is a moving target and no book can hope to capture a dynamic beast in a timeless manner. Nonetheless, the basics and core internals of the kernel are mature and I work hard to present the material with an eye to the future and with as wide applicability as possible.

Audience

This book targets software developers who are interested in understanding the Linux kernel. It is *not* a line-by-line commentary of the kernel source. Nor is it a guide to developing drivers or a reference on the kernel API (as if there even were a formal kernel API—hah!). Instead, the goal of this book is to provide enough information on the design and implementation of the Linux kernel that a sufficiently accomplished programmer can begin developing code in the kernel. Kernel development can be fun and rewarding, and I want to introduce the reader to that world as readily as possible. This book, however, in discussing both theory and application, should appeal to readers of either interest. I have always been of the mind that one needs to understand the theory to understand the application, but I do not feel that this book leans too far in either direction. I hope that whatever your motivations for understanding the Linux kernel, this book will explain the design and implementation sufficiently for your needs.

Thus, this book covers both the usage of core kernel systems and their design and implementation. I think this is important, and deserves a moment's discussion. A good example is Chapter 7, "Bottom Halves and Deferring Work," which covers bottom halves. In that chapter, I discuss both the design and implementation of the kernel's bottom-half mechanisms (which a core kernel developer might find interesting) and how to actually use the exported interfaces to implement your own bottom half (which a device driver developer might find interesting). In fact, I believe both parties should find both discussions relevant. The core kernel developer, who certainly needs to understand the inner workings of the kernel, should have a good understanding of how the interfaces are actually used. At the same time, a device driver writer will benefit from a good understanding of the implementation behind the interface.

This is akin to learning some library's API versus studying the actual implementation of the library. At first glance, an application programmer needs only to understand the API—it is often taught to treat interfaces as a black box, in fact. Likewise, a library developer is concerned only with the library's design and implementation. I believe, however, both parties should invest time in learning the other half. An application programmer who better understands the underlying operating system can make much greater use of it. Similarly, the library developer should not grow out of touch with the reality and practicality of the applications that use the library. Consequently, I discuss both the design and usage of kernel subsystems, not only in hopes that this book will be useful to either party, but also in hopes that the *whole* book is useful to both parties.

I assume that the reader knows the C programming language and is familiar with Linux. Some experience with operating system design and related computer science concepts is beneficial, but I try to explain concepts as much as possible—if not, there are some excellent books on operating system design referenced in the bibliography.

This book is appropriate for an undergraduate course introducing operating system design as the *applied* text if an introductory book on theory accompanies it. It should fare well either in an advanced undergraduate course or in a graduate-level course without ancillary material. I encourage potential instructors to contact me; I am eager to help.

Book Website

I maintain a website at http://tech9.net/rml/kernel_book/ that contains information pertaining to the book, including errata, expanded and revised topics, and information on future printings and editions. I encourage readers to check it out. I also apologize profusely for the previous end-of-sentence preposition, it was uncalled for, but the revamped sentence was hard to read, it was confusing, and *you deserve better.*

Second Edition Acknowledgments

Like most authors, I did not write this book in a cave (which is a good thing, because there are bears in caves) and consequently many hearts and minds contributed to the completion of this manuscript. Although no list would be complete, it is my sincere pleasure to acknowledge the assistance of many friends and colleagues who provided encouragement, knowledge, and constructive criticism.

First off, I would like to thank all of the editors who worked long and hard to make this book better. I would particularly like to thank Scott Meyers, my acquisition editor, for spearheading this second edition from conception to final product. I had the wonderful pleasure of again working with George Nedeff, production editor, who kept everything in order. Extra special thanks to my copy editor, Margo Catts. We can all only hope that our command of the kernel is as good as her command of the written word.

A special thanks to my technical editors on this edition: Adam Belay, Martin Pool, and Chris Rivera. Their insight and corrections improved this book immeasurably. Despite their sterling efforts, however, any remaining mistakes are my own fault. The same big thanks to Zack Brown, whose awesome technical editing efforts on the first edition still resonate loudly.

Many fellow kernel developers answered questions, provided support, or simply wrote code interesting enough on which to write a book. They are Andrea Arcangeli, Alan Cox, Greg Kroah-Hartman, Daniel Phillips, Dave Miller, Patrick Mochel, Andrew Morton, Zwane Mwaikambo, Nick Piggin, and Linus Torvalds. Special thanks to the kernel cabal (there is no cabal).

Respect and love to Paul Amici, Scott Anderson, Mike Babbitt, Keith Barbag, Dave Camp, Dave Eggers, Richard Erickson, Nat Friedman, Dustin Hall, Joyce Hawkins, Miguel de Icaza, Jimmy Krehl, Patrick LeClair, Doris Love, Jonathan Love, Linda Love, Randy O'Dowd, Sal Ribaudo and mother, Chris Rivera, Joey Shaw, Jon Stewart, Jeremy VanDoren and family, Luis Villa, Steve Weisberg and family, and Helen Whisnant.

Finally, thank you to my parents, for so much.

Happy Hacking!

Robert Love
Cambridge,
Massachusetts

About the Author

Robert Love is an open source hacker who has used Linux since the early days. Robert is active in and passionate about both the Linux kernel and the GNOME communities. Robert currently works as Senior Kernel Engineer in the Ximian Desktop Group at Novell. Before that, he was a kernel engineer at MontaVista Software.

Robert's kernel projects include the preemptive kernel, the process scheduler, the kernel events layer, VM enhancements, and multiprocessing improvements. He is the author and maintainer of *schedutils* and *GNOME Volume Manager.*

Robert has given numerous talks on and has written multiple articles about the Linux kernel. He is a Contributing Editor for *Linux Journal.*

Robert received a B.A. in Mathematics and a B.S. in Computer Science from the University of Florida. Born in South Florida, Robert currently calls Cambridge, Massachusetts home. He enjoys college football, photography, and cooking.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, in what areas you'd like to see us publish, and any other words of wisdom you're willing to pass our way.

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book and that due to the high volume of mail I receive I may not be able to reply to every message. When you write, please be sure to include this book's title and author as well as your name and email address or phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email:	feedback@novellpress.com
Mail:	Mark Taber Associate Publisher Novell Press/Pearson Education 800 East 96th Street Indianapolis, IN 46240 USA

Reader Services

For more information about this book or other Novell Press titles, visit our website at www.novellpress.com. Type the ISBN or the title of a book in the Search field to find the page you're looking for.

Table of Contents